Generating realistic content for benchmarking storage systems

João Paulo
jtpaulo@di.uminho.pt
University of Minho and INESC TEC

Universidade do Minho

December 15, 2023

Content-aware storage systems

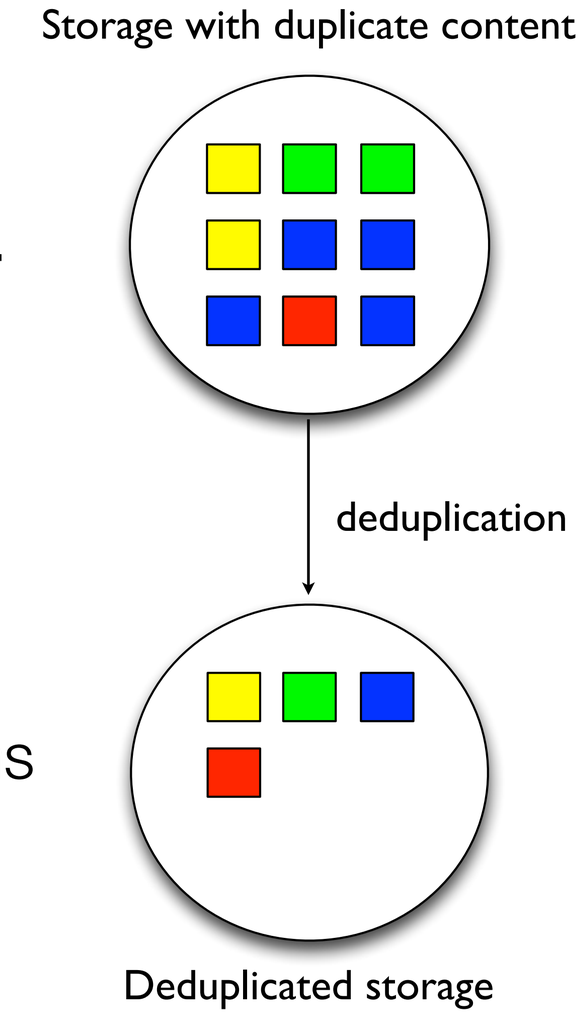
Context

- Storage systems implement different content-aware features
  - Data deduplication, compression, content-based caching, …

Content-aware storage systems

Context

- Storage systems implement different content-aware features
  - Data deduplication, compression, content-based caching, …
- Deduplication
  - Technique for eliminating duplicate data in storage systems
    - *e.g.*, fixed-sized blocks with identical content
    - interesting for VMs' virtual disks, block-devices, file systems



Benchmarking Deduplication Systems

Challenge

- Traditional disk-based I/O benchmarks (*e.g.*, IOzone, FIO, Bonnie++, VDBench)
  - Duplicated content generation patterns
    - **Single** - unrealistic high number of duplicates
    - **Random** - few to none duplicates
    - **Percentage** - fixed size number of groups with a similar number of duplicates

Benchmarking Deduplication Systems

Challenge

- Traditional disk-based I/O benchmarks (*e.g.*, IOzone, FIO, Bonnie++, VDBench)
  - Duplicated content generation patterns
    - **Singl**
    - **Random** - few to none duplicates
    - **Percentage** - fixed size number of groups with a similar number of duplicates

Unrealistic load for evaluating deduplication systems!

DEDISbench

Synthetic benchmark for block-based deduplication systems[1,2]

- Similar features as in other micro-benchmarks (*e.g.*, read and write operations, sequential and random access patterns,…)

[1] DEDISbench: A Benchmark for Deduplicated Storage Systems. DOA-SVI 2013

[2] Towards an Accurate Evaluation of Deduplicated Storage Systems. International Journal of Computer Systems Science and Engineering 2014

DEDISbench

Synthetic benchmark for block-based deduplication systems[1,2]

- Similar features as in other micro-benchmarks (*e.g.*, read and write operations, sequential and random access patterns,…)
- New features
  - Realistic distribution of duplicate blocks based on real datasets
  - Data integrity validation
  - Hotspot access pattern for I/O accesses

[1] DEDISbench: A Benchmark for Deduplicated Storage Systems. DOA-SVI 2013
[2] Towards an Accurate Evaluation of Deduplicated Storage Systems. International Journal of Computer Systems Science and Engineering 2014

DEDISbench

Synthetic benchmark for block-based deduplication systems[1,2]

- Similar features as in other micro-benchmarks (*e.g.*, read and write operations, sequential and random access patterns,…)
- New features
  - Realistic distribution of duplicate blocks based on real datasets
  - Data integrity validation
  - Hotspot access pattern for I/O accesses
- Available at https://github.com/jtpaulo/dedisbench
  - Being used by the community (*e.g.*, in storage deduplication papers)

[1] DEDISbench: A Benchmark for Deduplicated Storage Systems. DOA-SVI 2013
[2] Towards an Accurate Evaluation of Deduplicated Storage Systems. International Journal of Computer Systems Science and Engineering 2014

DEDISbench

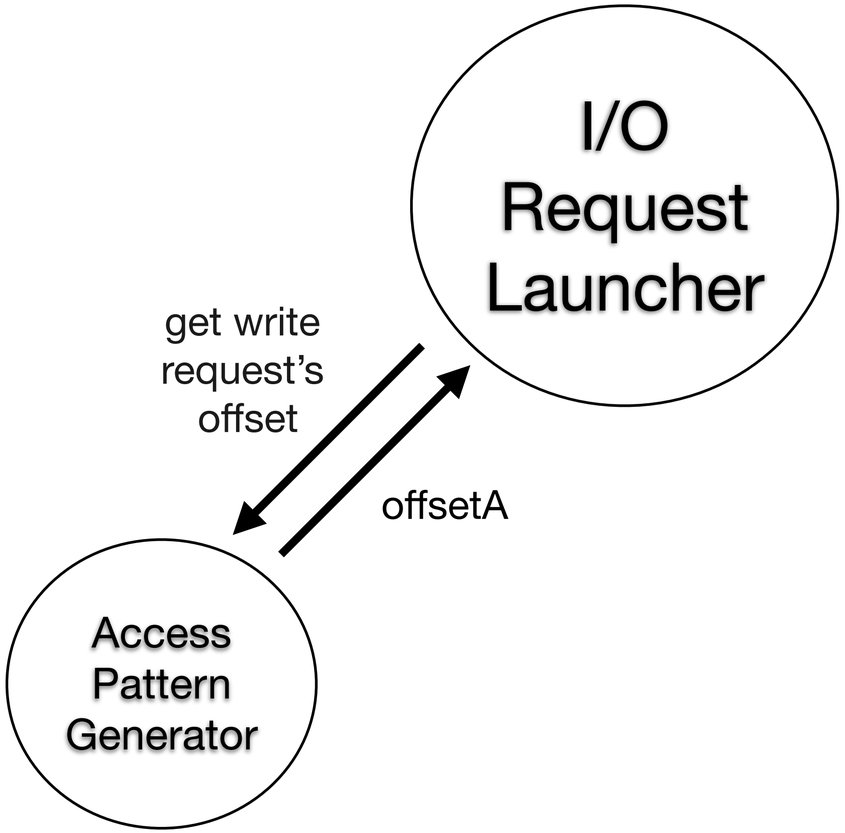
Runtime workflow

User command: DEDISbench -p -w -t5

Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes

DEDISbench

Runtime workflow

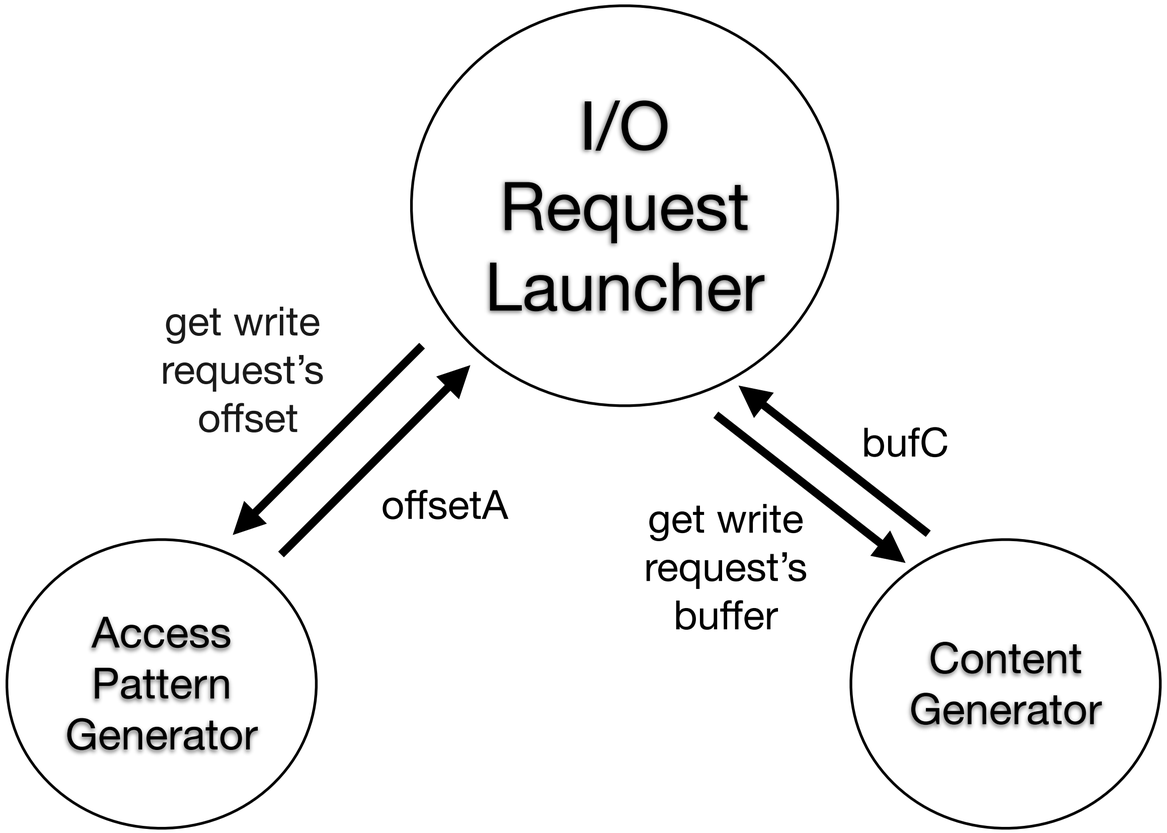


User command: DEDISbench -p -w -t5

Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes

DEDISbench

Runtime workflow



User command: DEDISbench -p -w -t5

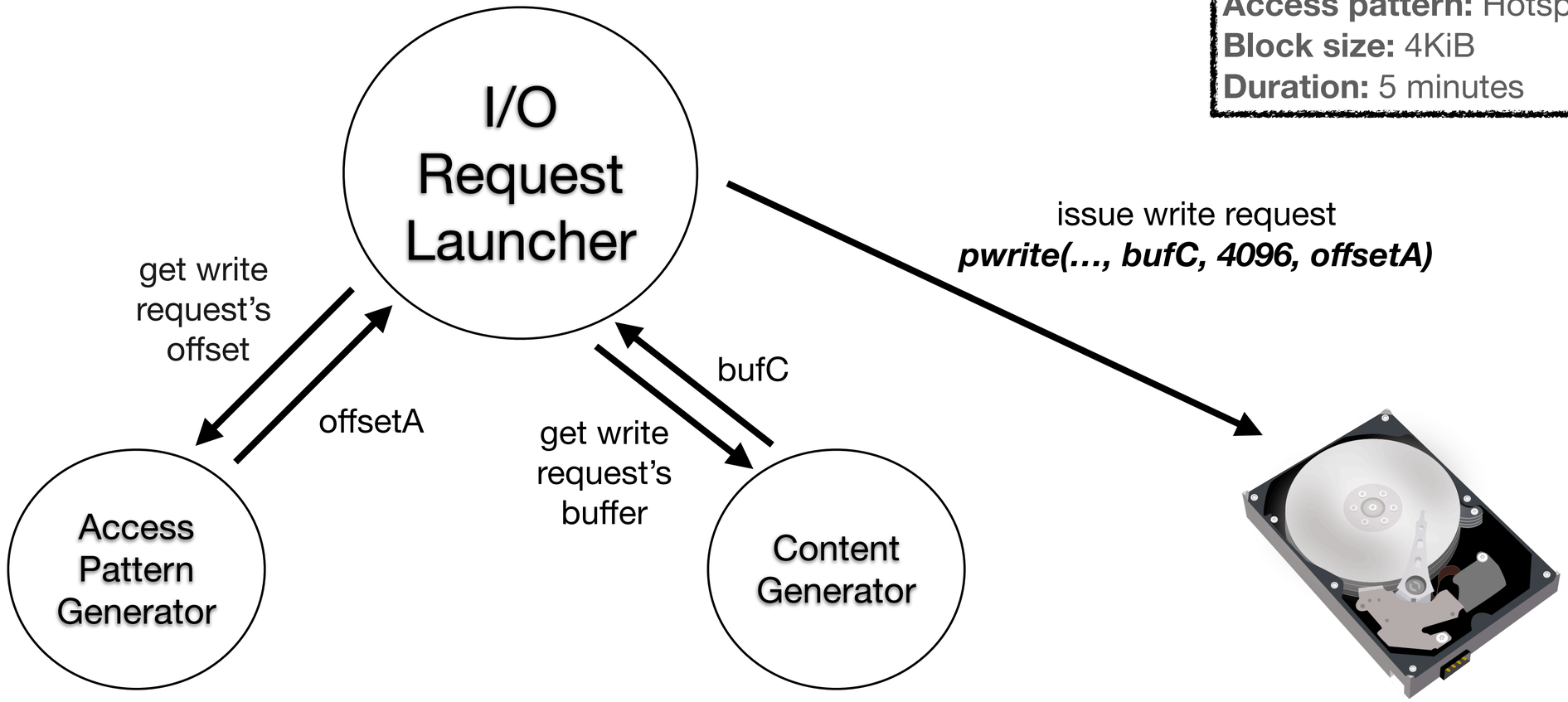
Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes

DEDISbench

Runtime workflow

User command: DEDISbench -p -w -t5

Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes

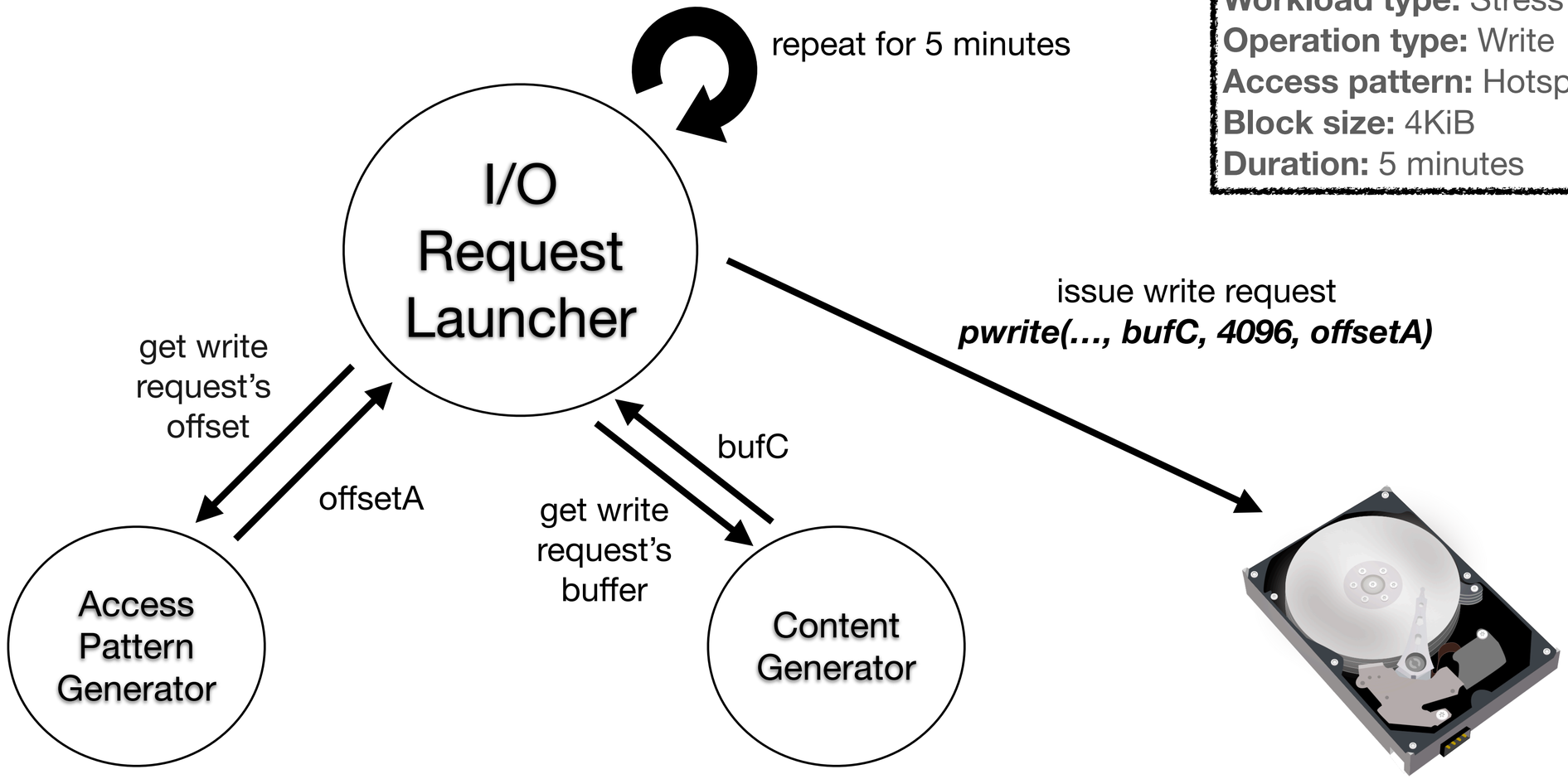


DEDISbench

Runtime workflow

User command: DEDISbench -p -w -t5

Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes

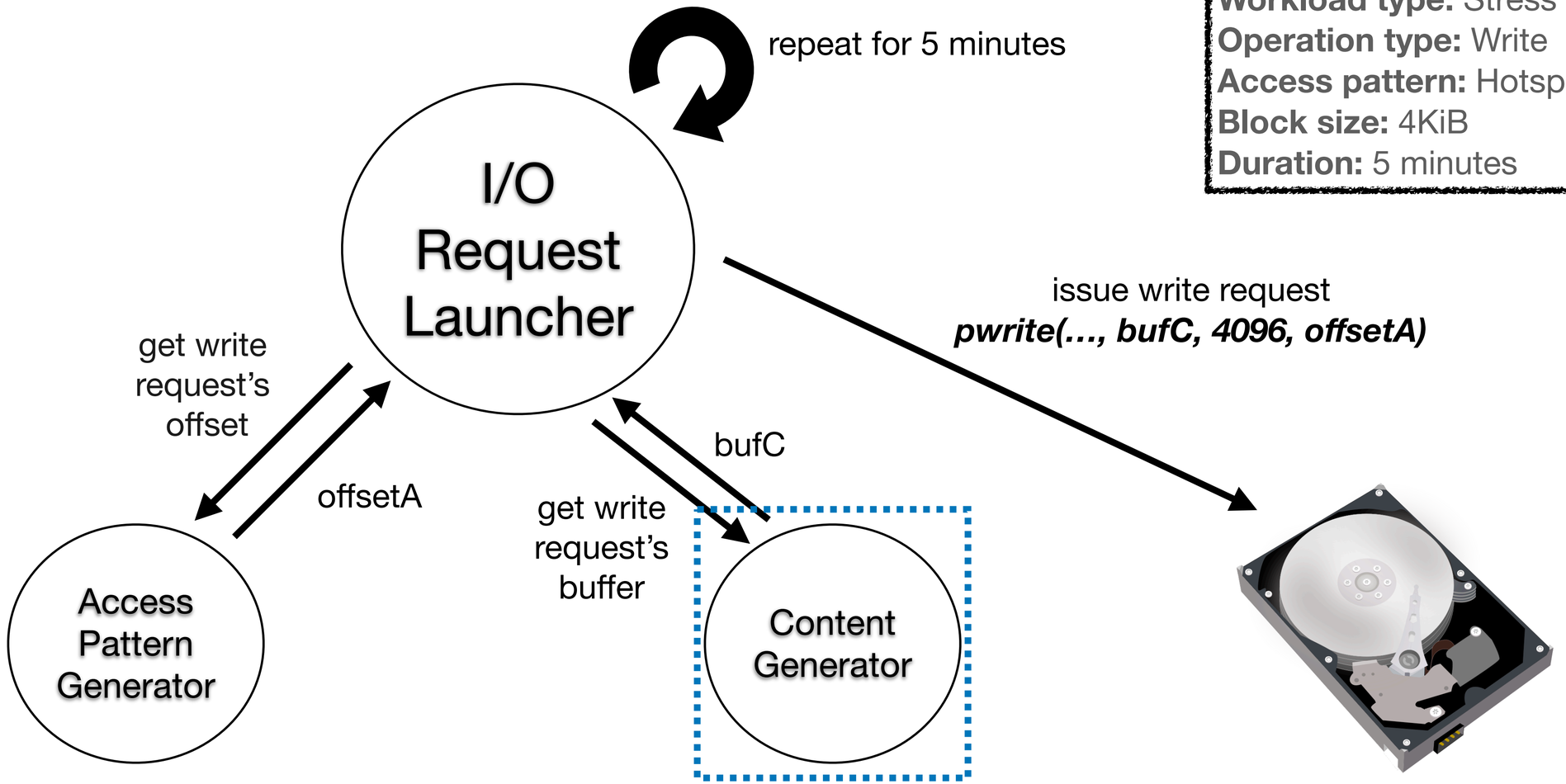


DEDISbench

Runtime workflow

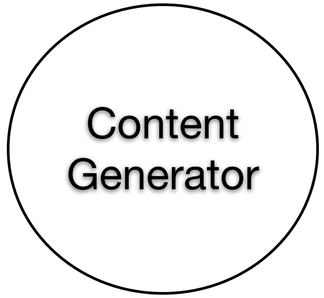
User command: DEDISbench -p -w -t5

Workload type: Stress testing
Operation type: Write
Access pattern: Hotspot
Block size: 4KiB
Duration: 5 minutes



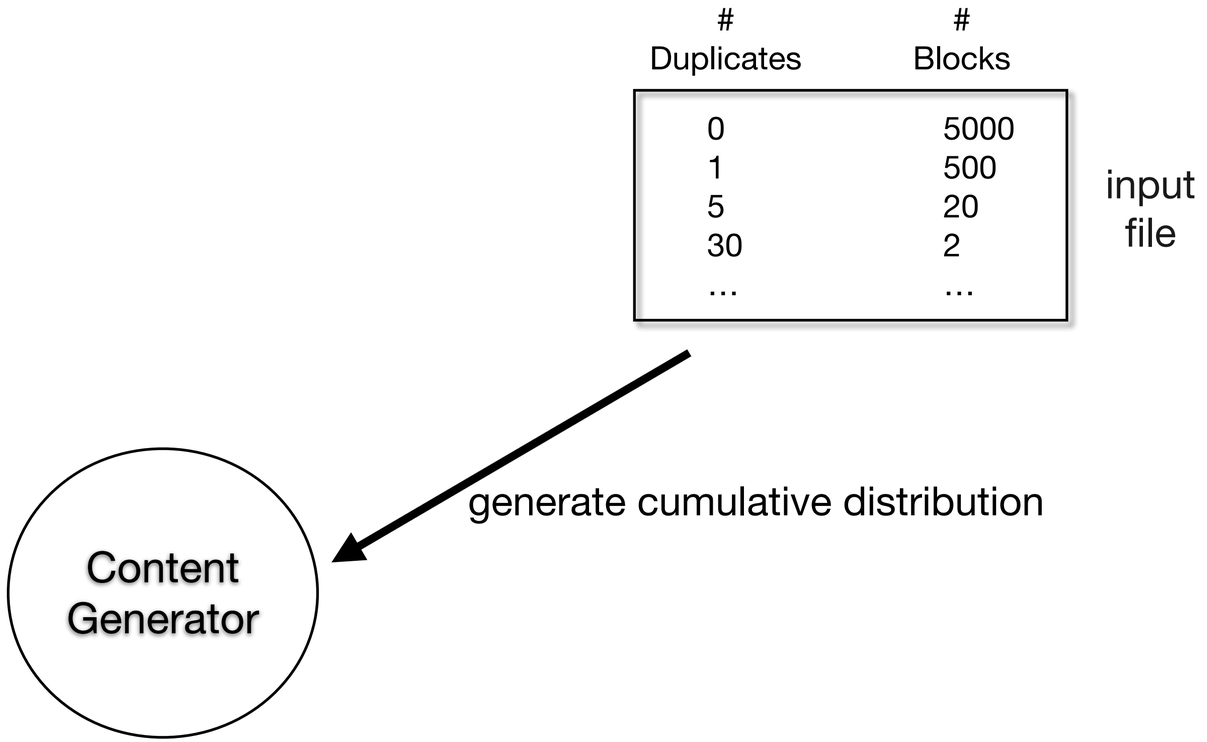
DEDISbench

Content generation and analysis



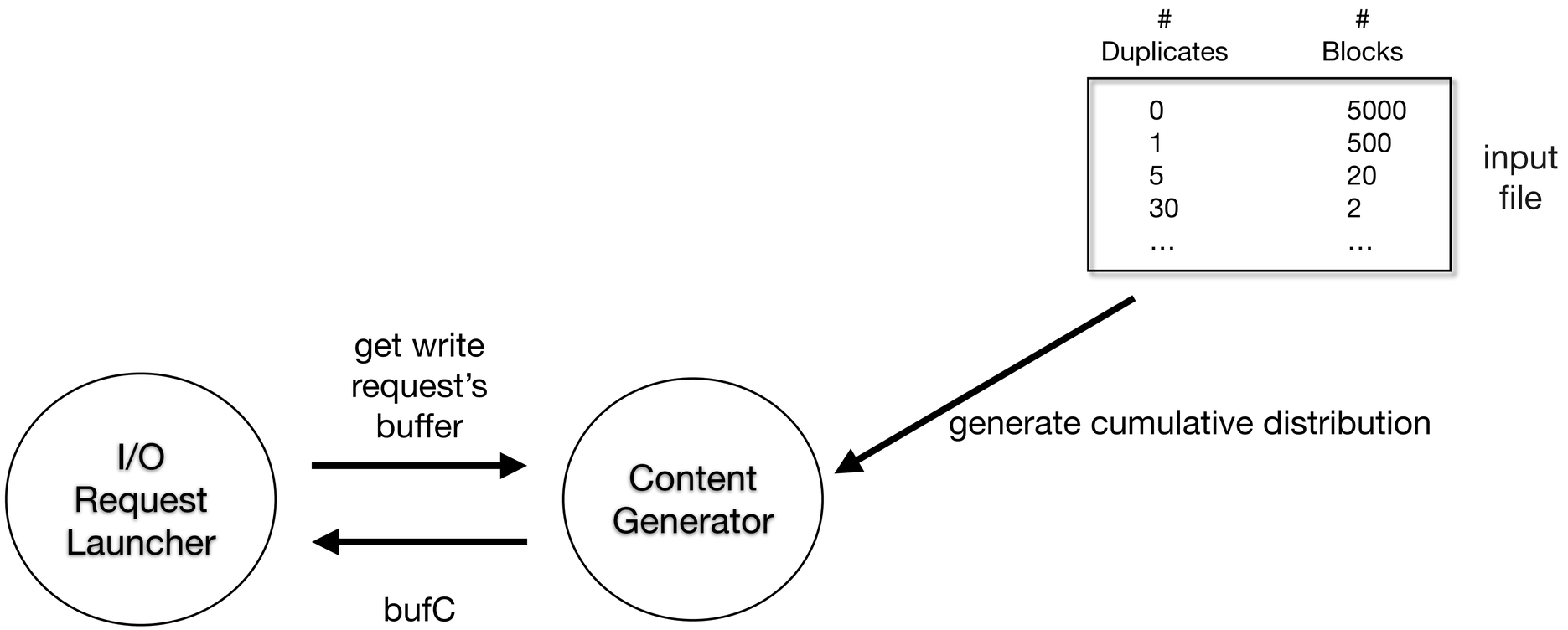
DEDISbench

Content generation and analysis



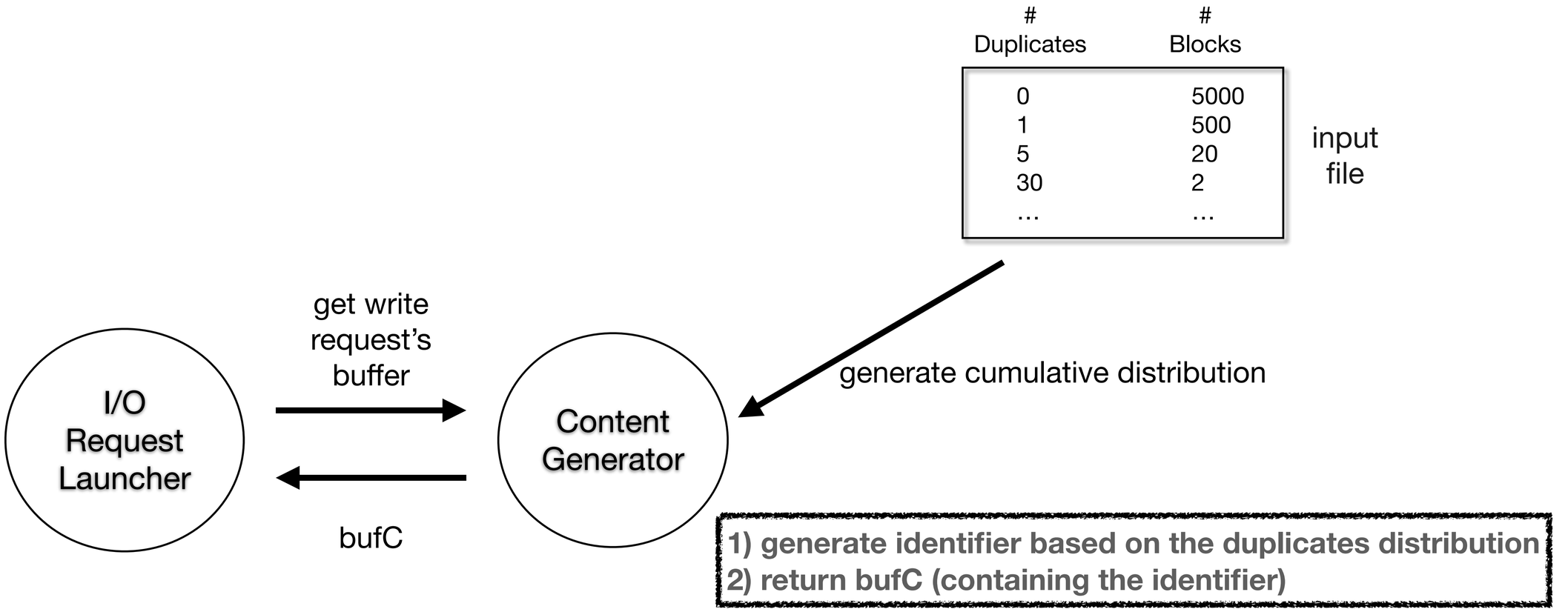
DEDISbench

Content generation and analysis



DEDISbench

Content generation and analysis



DEDISbench

Content generation and analysis

analyze dataset

DEDISgen

generate input file

# Duplicates	# Blocks
0	5000
1	500
5	20
30	2
...	...

input file

generate cumulative distribution

Content Generator

get write request's buffer

bufC

I/O Request Launcher

1) generate identifier based on the duplicates distribution
2) return bufC (containing the identifier)

DEDISbench

Content distributions

Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_highperf - storage appliance (31% of duplicates)
dist_kernels - linux kernels (61% duplicates)
dist_ubuntuamd - ubuntu cloud images (35% duplicates)

DEDISbench

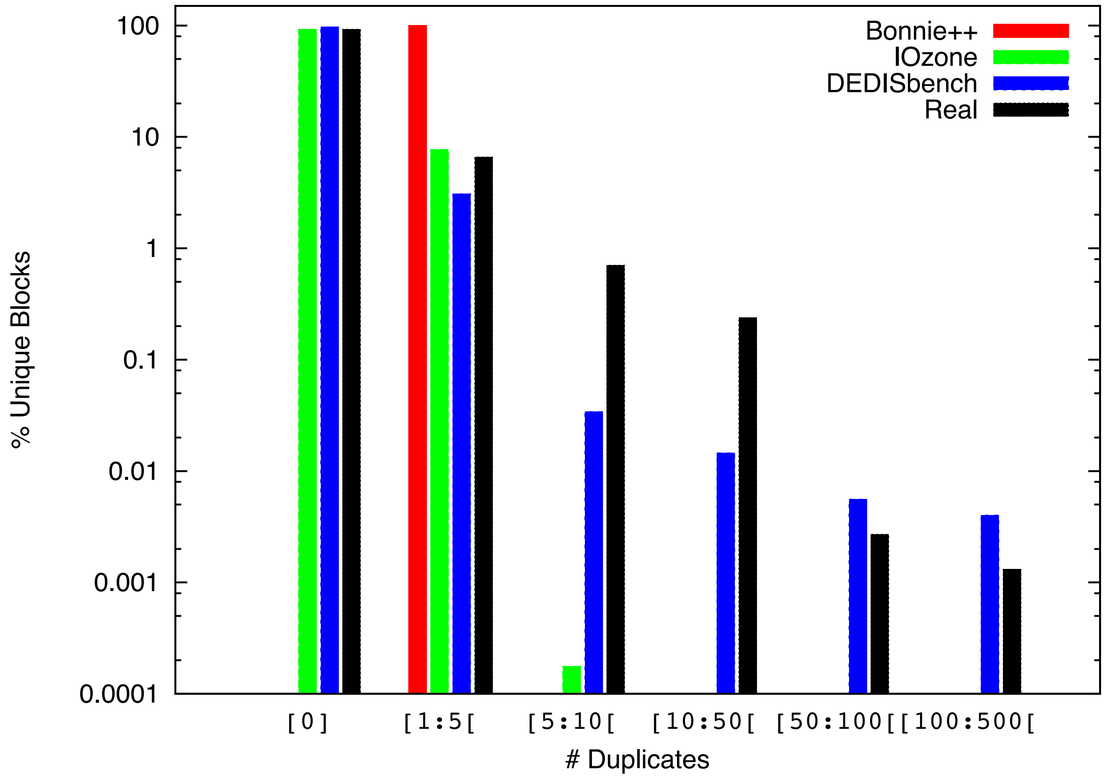
Content distributions

Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_highperf - storage appliance (31% of duplicates)
dist_kernels - linux kernels (61% duplicates)
dist_ubuntuamd - ubuntu cloud images (35% duplicates)

dist_personalfiles

Real - original dataset
Bonnie++ - does not allow specifying duplicates
IOzone - allows defining % of duplicates per group
DEDISbench - follows the distribution from the dataset



DEDISbench

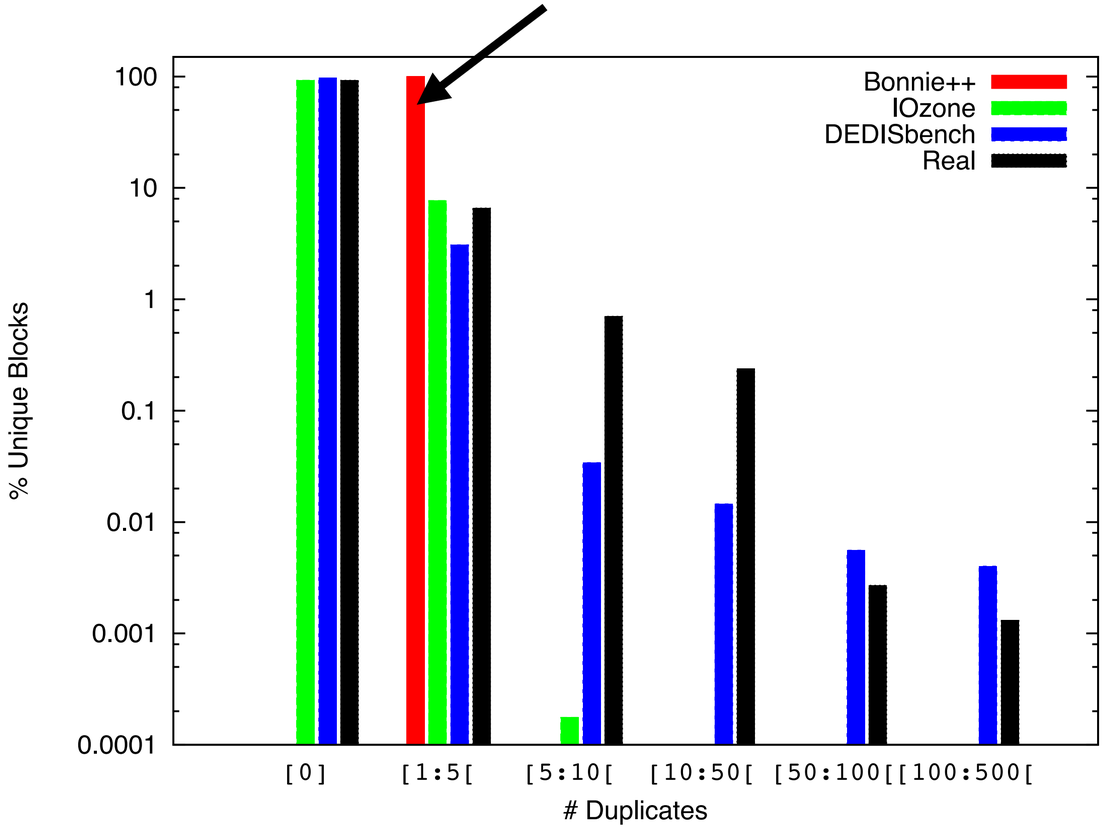
Content distributions

Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_highperf - storage appliance (31% of duplicates)
dist_kernels - linux kernels (61% duplicates)
dist_ubuntuamd - ubuntu cloud images (35% duplicates)

dist_personalfiles

Real - original dataset
Bonnie++ - does not allow specifying duplicates
IOzone - allows defining % of duplicates per group
DEDISbench - follows the distribution from the dataset



DEDISbench

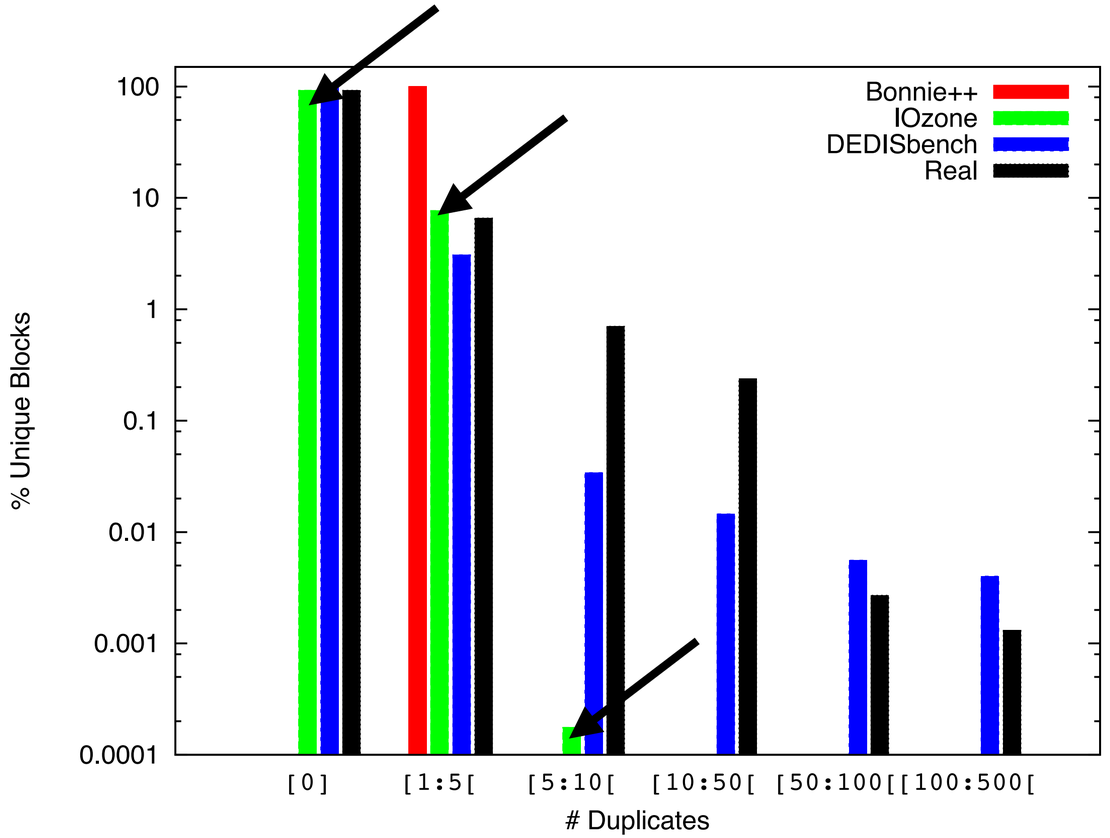
Content distributions

Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_highperf - storage appliance (31% of duplicates)
dist_kernels - linux kernels (61% duplicates)
dist_ubuntuamd - ubuntu cloud images (35% duplicates)

dist_personalfiles

Real - original dataset
Bonnie++ - does not allow specifying duplicates
IOzone - allows defining % of duplicates per group
DEDISbench - follows the distribution from the dataset



DEDISbench

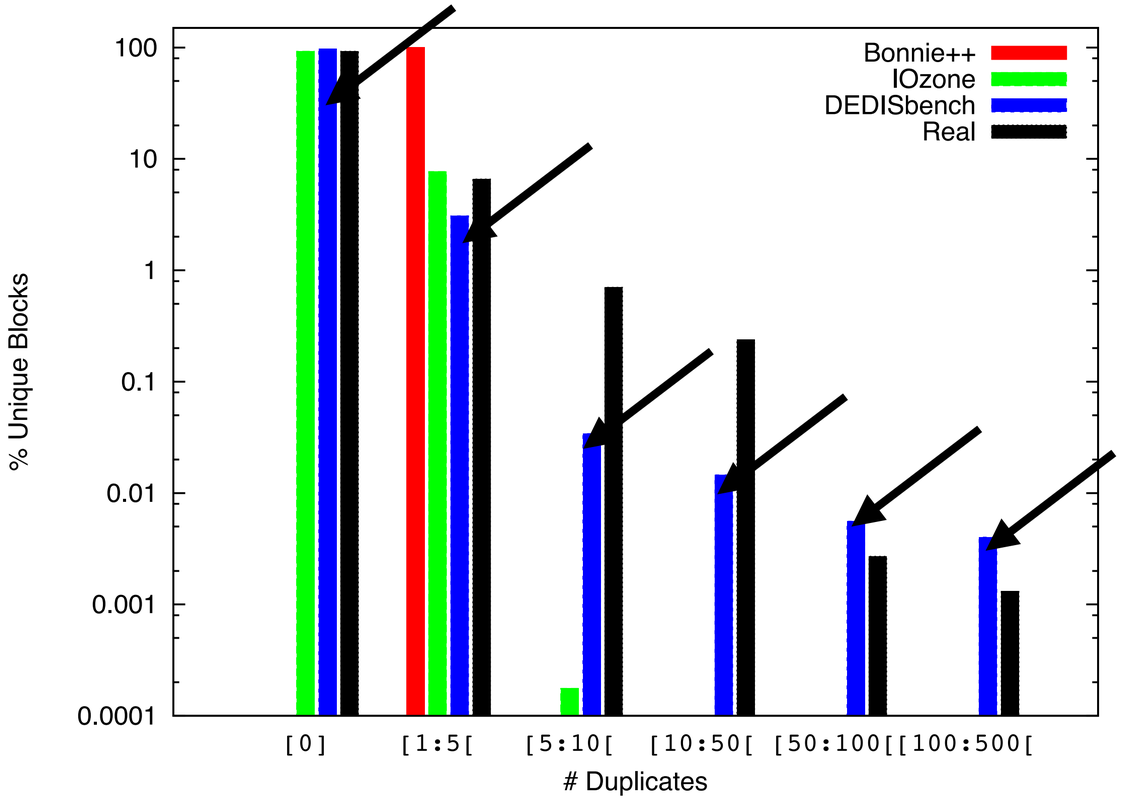
Content distributions

Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_highperf - storage appliance (31% of duplicates)
dist_kernels - linux kernels (61% duplicates)
dist_ubuntuamd - ubuntu cloud images (35% duplicates)

dist_personalfiles

Real - original dataset
Bonnie++ - does not allow specifying duplicates
IOzone - allows defining % of duplicates per group
DEDISbench - follows the distribution from the dataset



DEDISbench

Content distributions

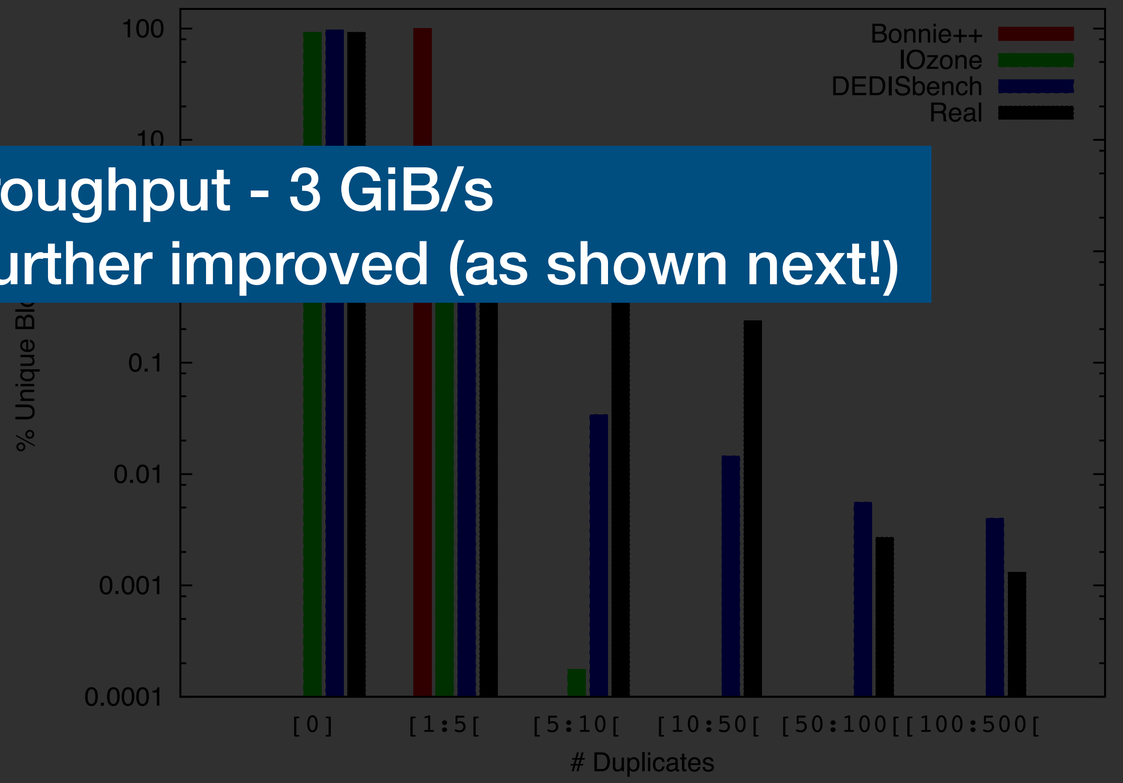
Supported distributions

dist_archival - archival storage (10% of duplicates)
dist_personalfiles - home folders (24% of duplicates)
dist_hig
dist_ker
dist_ubu

dist_personalfiles

Real - original dataset
Bonnie++ - does not allow specifying duplicates
IOzone - allows defining % of duplicates per group
DEDISbench - follows the distribution from the dataset

Block generation throughput - 3 GiB/s
Throughput and accuracy can be further improved (as shown next!)



Realistic Compression

Challenges and goals

- The content of each block is not realistic
  - Each block has an identifier (few bytes) plus static content
- What if the deduplication system compresses unique blocks?

Realistic Compression

Challenges and goals

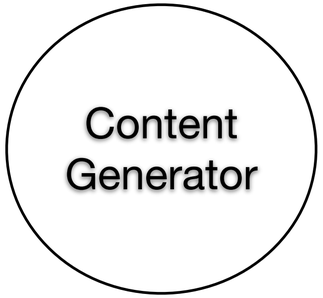
- The content of each block is not realistic
  - Each block has an identifier (few bytes) plus static content
- What if the deduplication system compresses unique blocks?
- Synthetic data should follow realistic compression factors
  - Intra-block - compressibility for each block
    - FIO, VDBench and IOZone only allow defining a single value for all blocks
  - Inter-block - compressibility across groups of blocks

Realistic Compressions

Work done so far

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file



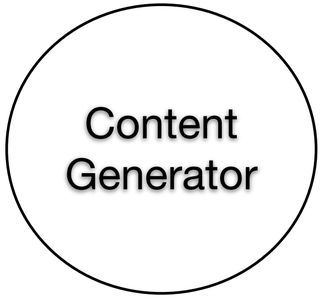
Realistic Compressions

Work done so far

50% of blocks are unique
19% of blocks have 5 copies

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file



Realistic Compressions

Work done so far

20% of these blocks compress from 10% to 19%
50% of these blocks compress from 30% to 39%

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file

Content Generator

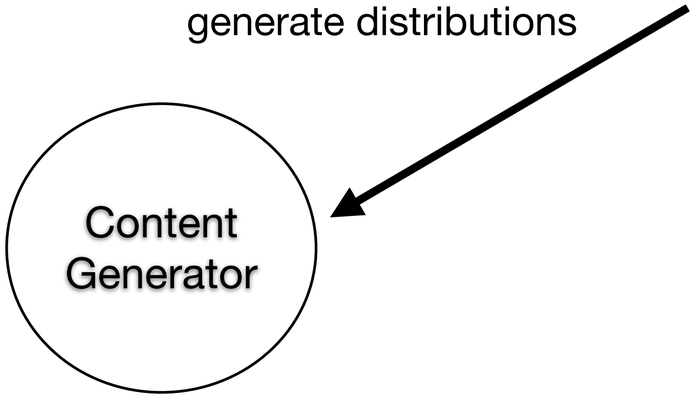
Realistic Compressions

Work done so far

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

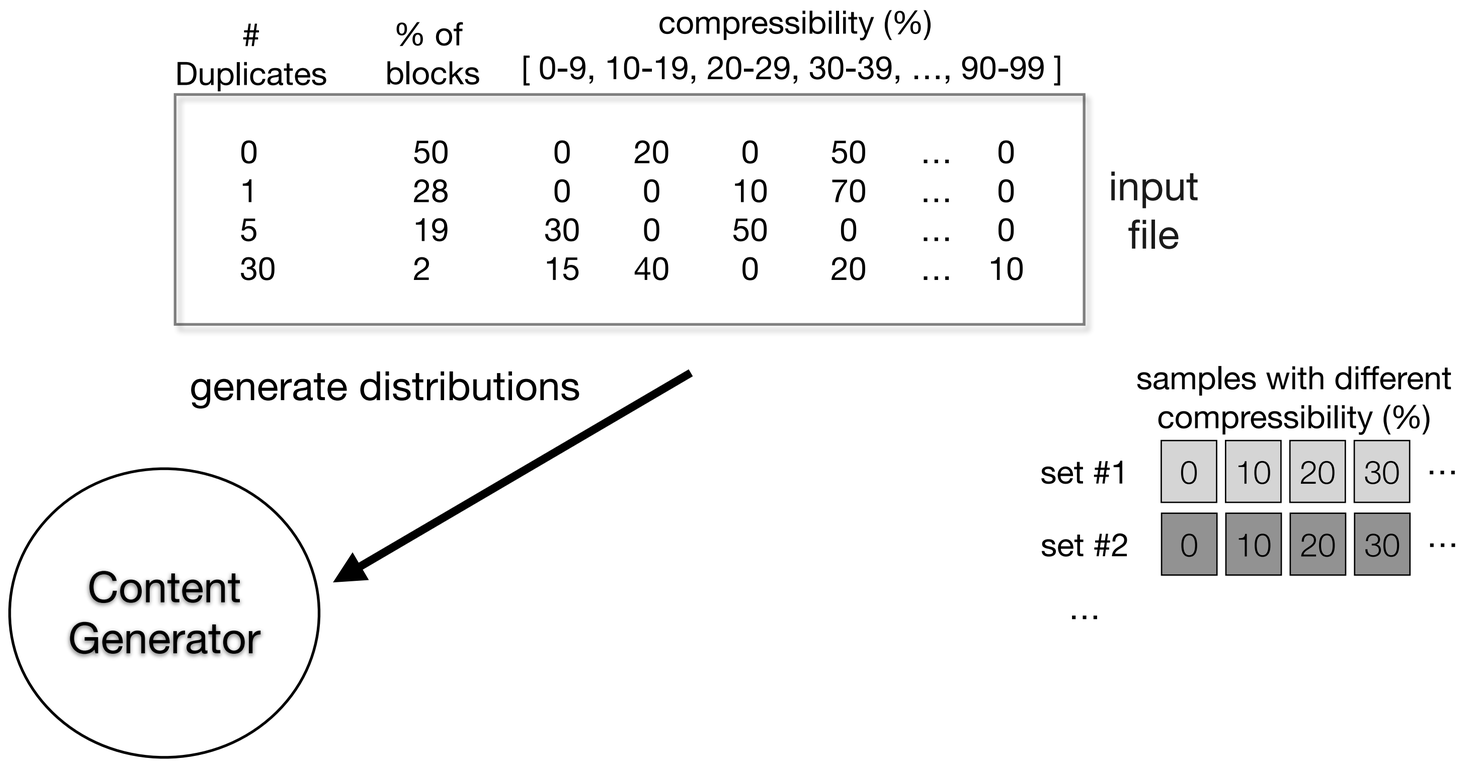
input file

generate distributions



Realistic Compressions

Work done so far



Realistic Compressions

Work done so far

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file

generate distributions

load samples into memory

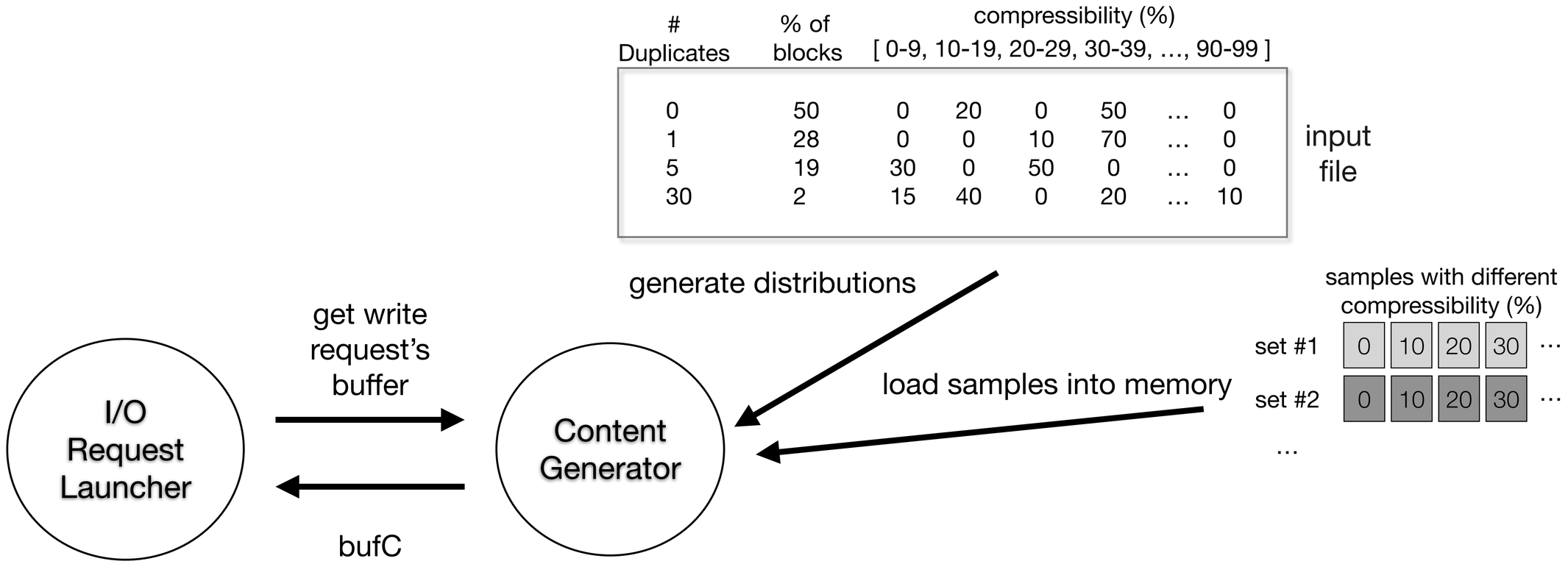
Content Generator

samples with different compressibility (%)

set #1	0	10	20	30	…
set #2	0	10	20	30	…
…					

Realistic Compressions

Work done so far

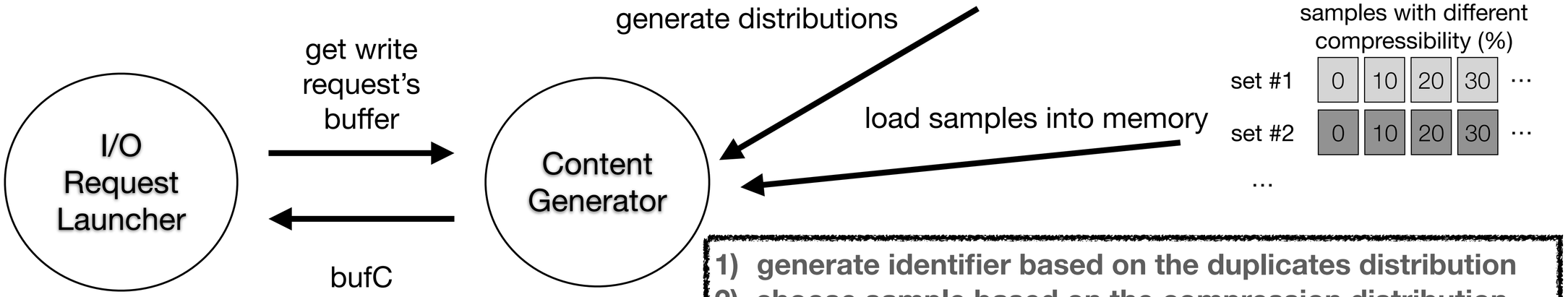


Realistic Compressions

Work done so far

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file



1) generate identifier based on the duplicates distribution
2) choose sample based on the compression distribution
3) return bufC (containing the identifier and sample content)

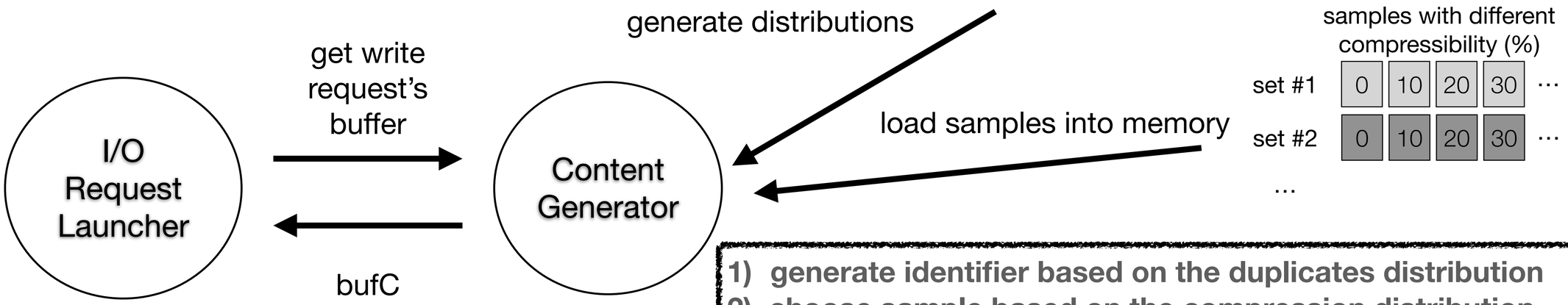
Realistic Compressions

Work done so far

Alternating between sets allows adjusting the average inter-block compressibility (can be defined by users)

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file



1) generate identifier based on the duplicates distribution
2) choose sample based on the compression distribution
3) return bufC (containing the identifier and sample content)

Realistic Compressions

Work done so far

Alternating between sets allows adjusting the average inter-block compressibility (can be defined by users)

generate input file

# Duplicates	% of blocks	compressibility (%) [0-9, 10-19, 20-29, 30-39, …, 90-99]					
0	50	0	20	0	50	…	0
1	28	0	0	10	70	…	0
5	19	30	0	50	0	…	0
30	2	15	40	0	20	…	10

input file

generate distributions

samples with different compressibility (%)

set #1	0	10	20	30	…
set #2	0	10	20	30	…
…					

load samples into memory

get write request's buffer

I/O Request Launcher

Content Generator

bufC

1) generate identifier based on the duplicates distribution
2) choose sample based on the compression distribution
3) return bufC (containing the identifier and sample content)

Realistic Compression

Work done so far

- Preliminary results - **dist_kernels**

Original dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	22,1	0,0	0,0	0,0	0,0	0,0	2,2	42,3	35,4	12,5	7,6
1-4	35,3	0,0	0,0	0,0	0,0	0,0	2,6	37,1	34,2	15,3	10,7
5-9	18,6	0,0	0,0	0,0	0,0	0,0	3,1	31,0	30,7	19,2	16,0
10-49	23,6	0,0	0,0	0,0	0,0	0,1	4,1	19,5	24,4	23,2	28,6
50-99	0,1	0,0	0,0	0,0	0,0	0,0	3,4	5,3	7,3	9,0	75,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,1	97,9

Synthetic dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	24,5	0,0	0,0	0,0	0,0	0,0	2,2	41,8	35,4	12,8	7,8
1-4	27,4	0,0	0,0	0,0	0,0	0,0	2,6	36,9	34,1	15,5	10,9
5-9	20,7	0,0	0,0	0,0	0,0	0,0	3,1	32,5	31,6	18,3	14,5
10-49	26,8	0,0	0,0	0,0	0,0	0,1	3,9	21,4	25,2	22,7	26,8
50-99	0,3	0,0	0,0	0,0	0,0	0,0	4,2	6,0	15,4	17,3	57,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,4	97,6

Realistic Compression

Work done so far

- Preliminary results - **dist_kernels**

Original dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	22,1	0,0	0,0	0,0	0,0	0,0	2,2	42,3	35,4	12,5	7,6
1-4	35,3	0,0	0,0	0,0	0,0	0,0	2,6	37,1	34,2	15,3	10,7
5-9	18,6	0,0	0,0	0,0	0,0	0,0	3,1	31,0	30,7	19,2	16,0
10-49	23,6	0,0	0,0	0,0	0,0	0,1	4,1	19,5	24,4	23,2	28,6
50-99	0,1	0,0	0,0	0,0	0,0	0,0	3,4	5,3	7,3	9,0	75,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,1	97,9

Synthetic dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	24,5	0,0	0,0	0,0	0,0	0,0	2,2	41,8	35,4	12,8	7,8
1-4	27,4	0,0	0,0	0,0	0,0	0,0	2,6	36,9	34,1	15,5	10,9
5-9	20,7	0,0	0,0	0,0	0,0	0,0	3,1	32,5	31,6	18,3	14,5
10-49	26,8	0,0	0,0	0,0	0,0	0,1	3,9	21,4	25,2	22,7	26,8
50-99	0,3	0,0	0,0	0,0	0,0	0,0	4,2	6,0	15,4	17,3	57,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,4	97,6

Distributions of duplicates are similar!

Realistic Compression

Work done so far

- Preliminary results - **dist_kernels**

Original dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	22,1	0,0	0,0	0,0	0,0	0,0	2,2	42,3	35,4	12,5	7,6
1-4	35,3	0,0	0,0	0,0	0,0	0,0	2,6	37,1	34,2	15,3	10,7
5-9	18,6	0,0	0,0	0,0	0,0	0,0	3,1	31,0	30,7	19,2	16,0
10-49	23,6	0,0	0,0	0,0	0,0	0,1	4,1	19,5	24,4	23,2	28,6
50-99	0,1	0,0	0,0	0,0	0,0	0,0	3,4	5,3	7,3	9,0	75,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,1	97,9

Synthetic dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	24,5	0,0	0,0	0,0	0,0	0,0	2,2	41,8	35,4	12,8	7,8
1-4	27,4	0,0	0,0	0,0	0,0	0,0	2,6	36,9	34,1	15,5	10,9
5-9	20,7	0,0	0,0	0,0	0,0	0,0	3,1	32,5	31,6	18,3	14,5
10-49	26,8	0,0	0,0	0,0	0,0	0,1	3,9	21,4	25,2	22,7	26,8
50-99	0,3	0,0	0,0	0,0	0,0	0,0	4,2	6,0	15,4	17,3	57,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,4	97,6

Distributions of compressibility are similar!

Realistic Compression

Work done so far

- Preliminary results - **dist_kernels**

Original dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	22,1	0,0	0,0	0,0	0,0	0,0	2,2	42,3	35,4	12,5	7,6
1-4	35,3	0,0	0,0	0,0	0,0	0,0	2,6	37,1	34,2	15,3	10,7
5-9	18,6	0,0	0,0	0,0	0,0	0,0	3,1	31,0	30,7	19,2	16,0
10-49	23,6	0,0	0,0	0,0	0,0	0,1	4,1	19,5	24,4	23,2	28,6
50-99	0,1	0,0	0,0	0,0	0,0	0,0	3,4	5,3	7,3	9,0	75,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,1	97,9

Average inter-block compression rate of 72.1%

Synthetic dataset

# Duplicates	% Blocks	Compressability % (separated by ranges)									
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90-99
0	24,5	0,0	0,0	0,0	0,0	0,0	2,2	41,8	35,4	12,8	7,8
1-4	27,4	0,0	0,0	0,0	0,0	0,0	2,6	36,9	34,1	15,5	10,9
5-9	20,7	0,0	0,0	0,0	0,0	0,0	3,1	32,5	31,6	18,3	14,5
10-49	26,8	0,0	0,0	0,0	0,0	0,1	3,9	21,4	25,2	22,7	26,8
50-99	0,3	0,0	0,0	0,0	0,0	0,0	4,2	6,0	15,4	17,3	57,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,4	97,6

Average inter-block compression rate of 75.5%

Realistic Compression

Work done so far

- Preliminary results - **dist_kernels**

Original dataset

# Duplicates	% Blocks	0-9									
0	22,1	0,0									
1-4	35,3	0,0	0,0	0,0	0,0	0,0	2,6	37,1	34,2	15,3	10,7
5-9	18,6	0,0	0,0	0,0	0,0	0,0	3,1	31,0	30,7	19,2	16,0
10-49	23,6	0,0	0,0	0,0	0,0	0,1	4,1	19,5	24,4	23,2	28,6
50-99	0,1	0,0	0,0	0,0	0,0	0,0	3,4	5,3	7,3	9,0	75,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,1	97,9

Block generation throughput - 23 GiB/s
Better content generation accuracy!

Synthetic dataset

											parated by ranges)
										-59	60-69 70-79 80-89 90-99
										,2	41,8 35,4 12,8 7,8
1-4	27,4	0,0	0,0	0,0	0,0	0,0	2,6	36,9	34,1	15,5	10,9
5-9	20,7	0,0	0,0	0,0	0,0	0,0	3,1	32,5	31,6	18,3	14,5
10-49	26,8	0,0	0,0	0,0	0,0	0,1	3,9	21,4	25,2	22,7	26,8
50-99	0,3	0,0	0,0	0,0	0,0	0,0	4,2	6,0	15,4	17,3	57,0
100-499	0,2	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,4	97,6

Average inter-block compression rate of 72.1%

Average inter-block compression rate of 75.5%

Realistic Compression

Work to be done

- Implementation needs to be further tested
  - Are samples realistic for multiple compression algorithms?
  - Is the distribution accurate for “all” datasets?

[1] SDGen: Mimicking Datasets for Content Generation in Storage Benchmarks. USENIX FAST 2015

Realistic Compression

Work to be done

- Implementation needs to be further tested
  - Are samples realistic for multiple compression algorithms?
  - Is the distribution accurate for “all” datasets?
- Comparison with other alternatives
  - SDGen[1] - https://github.com/iostackproject/SDGen
  - Accuracy vs performance
    - SDGen is more accurate but content generation is slower (~100 MiB/s)

[1] SDGen: Mimicking Datasets for Content Generation in Storage Benchmarks. USENIX FAST 2015

DSR - Distributed Storage Research

Team

- Storage for DL
  - **Monarch[1]:** transparent storage tiering for DL
- Storage Diagnosis and Benchmarking
  - **DIO[2]:** diagnosing applications' I/O behavior
- Website: https://dsr-haslab.github.io
- Github: https://github.com/dsrhaslab

[1] Accelerating Deep Learning Training Through Transparent Storage Tiering. ACM/IEEE CCGrid 2022
[2] Toward a practical and timely diagnosis of applications' I/O behavior. IEEE Access 2023