Hendrik Nolte, Dorothea Sommer,Julian Kunkel

Using GPUDirect Storage

6.12.2023

GPUDirect Storage Overview

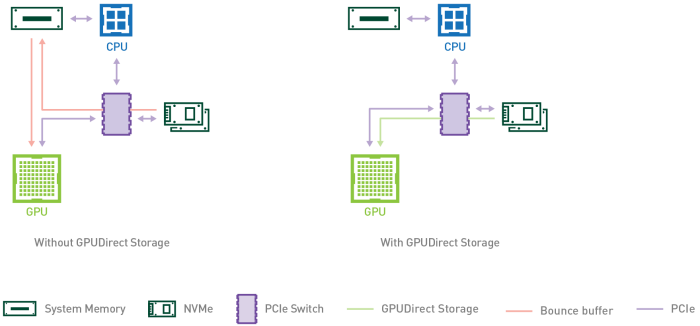


Abbildung: https://developer.nvidia.com/blog/gpudirect-storage/

NVidia Data Loading Library (DALI)

- Drop-in replacement for data loaders/iterators
- Move preprocessing also to GPU
  - loading
  - decoding
  - cropping
  - resizing
- Works with Tensorflow, PyTorch, ...

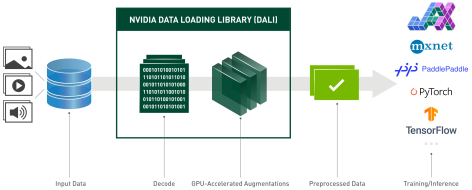


Abbildung: `https://github.com/NVIDIA/DALI`

DALI Example

- Reads data directly from storage
- loss, model and back propagation need to be implemented using
  - Tensorflow
  - PyTorch
- Can return Tensorflow tenor tuple
- Different compatibilities, e.g.
  - tfRecord

```python
@pipeline_def(num_threads=4, device_id=0)
def get_dali_pipeline():
    images, labels = fn.readers.file(
        file_root=images_dir, random_shuffle=True, name="Reader")
    # decode data on the GPU
    images = fn.decoders.image_random_crop(
        images, device="mixed", output_type=types.RGB)
    # the rest of processing happens on the GPU as well
    images = fn.resize(images, resize_x=256, resize_y=256)
    images = fn.crop_mirror_normalize(
        images,
        crop_h=224,
        crop_w=224,
        mean=[0.485 * 255, 0.456 * 255, 0.406 * 255],
        std=[0.229 * 255, 0.224 * 255, 0.225 * 255],
        mirror=fn.random.coin_flip())
    return images, labels


train_data = DALIGenericIterator(
    [get_dali_pipeline(batch_size=16)],
    ['data', 'label'],
    reader_name='Reader'
)


for i, data in enumerate(train_data):
    x, y = data[0]['data'], data[0]['label']
    pred = model(x)
    loss = loss_func(pred, y)
    backward(loss, model)
```

Abbildung: https://github.com/NVIDIA/DALI

IO500

- IO500 normally allocates the buffer on CPU only
- IO500 uses currently the timestamp pattern
- Phase-concurrent branch includes options to trigger the benchmarks:
  - allocateBufferDevice
  - gpuDirect
  - The options work as described for the benchmark repositories
- Also includes concurrent phase - runs phases at the same time
  - IOR easy write (20%)
  - RND 1 MB read (40%)
  - MDWorkbench (40%)
- Setting the flags, triggers the options for ALL phases and all benchmarks

Benchmarks

- Core benchmarks IOR/MDTest/MDWorkbench support GPUDirect
  - Normally, IO is done between Client NIC + (host) memory
  - GPUDirect: IO is done between Client NIC + GPU memory - skipping host mem
- We can choose if data buffers and patterns are created/verified on GPU/CPU
- Extra flag: allocateBufferOnGPU=MODE

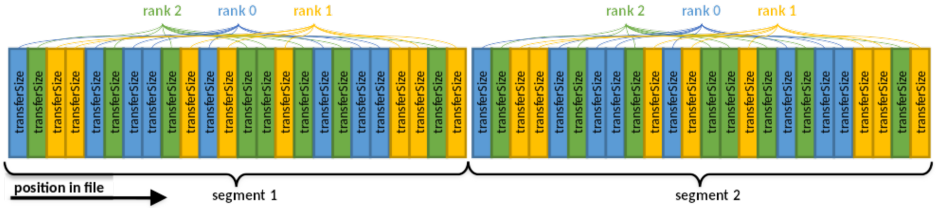
Mode	Buffer	Creation, Verification (if enabled)	GPUDirect
0	malloc()	CPU	No
1	cudaMallocManaged()	CPU	Optional
2	cudaMallocManaged()	GPU	Optional
3	cudaMalloc()	GPU	Mandatory

- To enable GPUDirect: –gpuDirect
- Requires: POSIX odirect
  GPUDirect supports unaligned blocks (with performance impact)
- Limitations: Verification is supported currently only for timestamp pattern

Mode Extended

- `phase-concurrent` branch also includes the `-mode=extended` option
- introduces new operations like concurrent
  - Default: 20% do write, 40% do reads, 40% do metadata
- And random 4k/1MiB write/reads

Transfers in one segment are randomized, the same pattern is repeated across segments



Hardware

- IO500 was run on Grete
  - https://www.top500.org/system/180092/
  - CPU: AMD EPYC 7513 32C
  - Interconnect: Infiniband HDR
  - Accelerator: 4xA100 SXM4 80GB
  - Storage: DDN Lustre 130 TiB NVME

Preliminary Results

Task / Mode	0	1	2	2-GPUD	3-GPUD
ior-easy-write [GiB/s]	6.3	7.5	7.0	6.1	5.6
ior-rnd4K-write [GiB/s]	0.2	0.2	0.19	0.02	0.02
mdtest-easy-write [kIOPS]	11.7	11.5	11.5	8.4	8.3
ior-rnd1MB-write [GiB/s]	1.2	0.9	1.2	5.2	3.8
mdworkbench-create	11.0	11.0	11.0	3.4	3.3
find-easy [kIOPS]	2635.5	2219.4	2501.2	2241.7	2433.7
ior-hard-write [GiB/s]	0.7	0.4	0.4	0.1	0.1
mdtest-hard-write [kIOPS]	2.5	2.8	2.8	2.5	2.3
find [kIOPS]	1577.8	1543.6	1473.7	2713.4	2624.0
ior-rnd4K-read [GiB/s]	2.4	0.1	0.2	0.03	0.03
ior-rnd1MB-read [GiB/s]	26.9	2.8	3.3	5.2	4.2
find-hard [kIOPS]	1364.6	1655.5	1398.9	1342.5	1201.2
mdworkbench-bench [kIOPS]	18.6	18.3	3.1	8.4	2.9
concurrent [score]	6.5	6.3	3.6	7.7	4.9
ior-easy-read [GiB/s]	5.8	6.1	3.6	6.2	6.1
mdtest-easy-stat [kIOPS]	28.8	28.7	29.6	207.8	200.0
ior-hard-read [GiB/s]	3.0	2.2	0.24	0.3	0.3
mdtest-hard-stat [kIOPS]	49.5	49.7	46.4	194.0	190.8
mdworkbench-find-delete [kIOPS]	19.9	19.9	20.8	19.9	20.1
mdtest-easy-delete [kIOPS]	21.8	22.3	19.7	22.0	20.0
mdtest-hard-read [kIOPS]	14.4	14.4	4.9	5.0	5.0
mdtest-hard-delete [kIOPS]	5.0	4.9	4.9	5.1	4.7
Score Bandwidth [GiB/s]	2.9	2.5	1.2	1.0	1.0
Score IOPS [kIOPS]	23.8	24.1	20.4	32.6	31.2
ScoreX Bandwidth [GiB/s]	2.4	1.1	0.6	0.6	0.5
ScoreX IOPS [kIOPS]	47.6	48.0	37.1	54.2	48.0

- Number shows the mode
- GPUDirect on/off
- Used a single node on Grete
  - 9 processes
  - 3 GPUs
- Numbers are irrelevant
  $\rightarrow$ Just want to show it works
  $\rightarrow$ And how it looks like
- Shows volatility to file count
  slowdown due md create good perf
- Find/Easy hard consistent results, not find.

Have a Look at the IO500 List

- Check out the IO500 list
  - ▶ https://io500.org/

Production | 10 Node Production | **Research** | 10 Node Research | Full | Historical

Ranking of the research system submissions. This is a subset of the Full List of submissions, showing only one highest-scoring result per storage system. This list also contains all valid IO500 submissions prior to the creation of the Research List.

#	INFORMATION						IO500			
	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	SCORE	BW (GiB/S)	MD (KIOP/S)	REPRO.
1	ISC23	Pengcheng Laboratory	Pengcheng Cloudbrain-II on Atlas 900	Pengcheng Laboratory and Tsinghua University	SuperFS	300	36,000	210,254.98	4,847.48	9,119,612.35	🔒
2	ISC23	JNIST and HUST PDSL	Cheeloo-1 with OceanStor Pacific	Huawei	OceanFS2	10	9,600	137,100.02	2,439.37	7,705,448.04	✓
3	SC22	Argonne National Laboratory	Aurora Storage	Intel	DAOS	260	27,040	20,694.50	6,048.69	70,802.51	-
4	SC22	Sugon Cloud Storage Laboratory	ParaStor	Sugon	ParaStor	10	2,560	8,726.42	718.11	106,042.93	-
5	SC22	SuPro Storteck	StarStor	SuPro Storteck	StarStor	10	2,560	6,751.75	515.15	88,491.65	-
6	SC22	Tsinghua Storage Research Group	SuperStore	Tsinghua Storage Research Group	SuperFS	10	1,200	5,517.73	179.60	169,515.95	-
7	ISC22	National Supercomputing Center in Jinan	Shanhe	PDSL	flashfs	10	2,560	3,534.42	207.79	60,119.50	-
8	SC22	Cloudam HPC on OCI	HPC-OCI	Cloudam	BurstFS	64	1,920	3,033.03	278.48	33,033.54	-
9	SC21	Huawei HPDA Lab	Athena	Huawei	OceanFS	10	1,720	2,395.03	314.56	18,235.71	-
10	SC21	Olympus Lab	OceanStor Pacific	Huawei	OceanFS	10	1,720	2,298.69	317.07	16,664.88	-
11	ISC23	LRZ	SuperMUC-NG-Phase2	Lenovo	DAOS	24	2,688	2,111.06	433.05	10,291.12	✓
12	SC21	Huawei Cloud		PDSL	Flashfs	15	1,560	2,016.70	109.82	37,034.00	-
13	ISC21	Intel	Endeavour	Intel	DAOS	10	1,440	1,859.56	398.77	8,671.65	-
14	ISC20	Intel	Wolf	Intel	DAOS	52	1,664	1,792.98	371.67	8,649.57	-
15	ISC22	University of Cambridge	Cumulus	Dell/Intel	DAOS	200	2,000	1,107.17	283.19	4,328.68	-
16	SC22	Meadowgate Technologies	Meadowgate	INTEL HPE	DAOS	10	1,280	1,014.24	213.15	4,826.12	-

How to Get Started

- We need your help!
- Please, test the features on your system
- If there are any issues → open an issue!
- Get Started:
  - git clone `https://github.com/IO500/io500`
  - git checkout `phase-concurrent`
  - set `IOR_HASH=db3c6fb` in `prepare.sh`
  - Ensure that you also have CUDA available