Best Performance

Table of contents

NVIDIA Parabricks software can give very high performance when all the required computing resources are provided to it. It should meet all the requirements in Installation Requirements section. Here are a few examples of how to get Parabricks software to give its best performance.

See the Hardware Requirements section for minimum hardware requirements.

See the Software Requirements section for minimum software requirements.

Basic Performance Tuning

The goal of the NVIDA Parabricks software is to get the highest performance for bioinformatics and genomic analysis. There are a few key, basic system options that you can tune to achieve maximum performance.

Use a Fast SSD

Parabricks software operates with two kinds of files:

- Input/output files specified by the user
- Temporary files created during execution and deleted at the end of the run

The best performance is achieved when both kinds of files are on a fast, local SSD. If this is not possible you can place the input/output files on a fast network storage device and the temporary files on a local SSD using the `--tmp-dir` option.

> **Note**
>
> Tests have shown that you can use up to 4 GPUs and still get good performance with the Lustre network for Input/Output files. If you

plan to use more than 4 GPUs, we highly recommend using local SSDs for all kinds of files.

DGX Users

The DGX comes with a SSD, usually mounted on `/raid`. Use this disk, and use a directory on this disk as the `--tmp-dir`. For initial testing, you can even copy the input files to this disk to eliminate variability in performance.

In certain cases, Transparent HugePage Support (THP) has been found to increase performance. Consider enabling THP and testing performance on benchmark cases.

Specifying which GPUs to use

You can choose the number of GPUs to run using the command line option `--num-gpus N` for those tools that use GPUs. With this option only the first `N` GPUs listed in the output of `nvidia-smi` will be used.

To use specific GPUs set the environment variable `NVIDIA_VISIBLE_DEVICES`. GPUs are numbered starting with zero. For example, this command will use only the second (GPU #1) and fourth (GPU #3) GPUs:

```
$ NVIDIA_VISIBLE_DEVICES="1,3" pbrun fq2bam --num-gpus 2 --ref Ref.fa --in-fq S1_1.fastq.gz --in-fq S1_2.fastq.gz
```

Tool-Specific Performance Guidelines

This section details guidelines specific to individual tools.

Best Performance for Germline Pipeline

On an H100 DGX the gremline pipeline typically runs in under ten minutes.

For backwards-compatible results with less performance when using `--fq2bamfast`, set `--bwa-options="-K 10000000"`.

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir \ --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun germline \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-fq /workdir/fastq1.gz /workdir/fastq2.gz \ --out-bam /outputdir/fq2bam_output.bam \ --tmp-dir /workdir \ --num-cpu-threads-per-stage 16 \ --bwa-cpu-thread-pool 16 \ --out-variants /outputdir/out.vcf \ --run-partition \ --read-from-tmp-dir \ --gpusort \ --gpuwrite \ --fq2bamfast \ --keep-tmp
```

Best Performance for Deepvariant Germline Pipeline

For backwards-compatible results with less performance when using `--fq2bamfast`, set `--bwa-options="-K 10000000"`.

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir \ --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun deepvariant_germline \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-fq /workdir/fastq1.gz /workdir/fastq2.gz \ --out-bam /outputdir/fq2bam_output.bam \ --tmp-dir /workdir \ --num-cpu-threads-per-stage 16 \ --bwa-cpu-thread-pool 16 \ --out-variants /outputdir/out.vcf \ --run-partition \ --read-from-tmp-dir \ --num-streams-per-gpu 4 \ --gpusort \ --gpuwrite \ --fq2bamfast \ --keep-tmp
```

Best Performance for PacBio Germline Pipeline

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun pacbio_germline \ --ref /workdir/${REFERENCE_FILE} \ --in-fq /workdir/${INPUT_FASTQ} \ --out-bam
```

```
/outputdir/${OUTPUT_BAM} \ --out-variants /outputdir/out.vcf --num-chaining-threads 3 \ --alignment-large-pair-size 5000 \ --process-large-alignments-on-cpu \ --num-alignment-threads-per-gpu 8 \ --num-alignment-device-mem-buffers 8 \ --run-partition \ --read-from-tmp-dir \ --num-streams-per-gpu 4 \ --gpusort \ --gpuwrite \ --keep-tmp
```

Best Performance for fq2bam/fq2bamfast

Use the new beta version, fq2bamfast. For backwards-compatible results with less performance, set `--bwa-options="-K 10000000"`.

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun fq2bamfast \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-fq /workdir/fastq1.gz /workdir/fastq2.gz \ --out-bam /outputdir/fq2bam_output.bam \ --tmp-dir /workdir \ --bwa-cpu-thread-pool 16 \ --out-recal-file recal.txt \ --knownSites /workdir/hg.known_indels.vcf \ --gpusort \ --gpuwrite
```

Best Performance for deepvariant

DeepVariant from Parabricks has the ability to use multiple streams on a GPU. The number of streams that can be used depends on the available resources. The default number of streams is set to two but can be increased up to a maximum of six to get better performance. This is something that has to be experimented with, before getting the optimal number on your system.

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun deepvariant \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-bam
```

```
/outputdir/fq2bam_output.bam \ --out-variants /outputdir/out.vcf \ --num-streams-per-gpu 4 \ --run-partition
```

Best Performance for haplotypecaller

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun haplotypecaller \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-bam /outputdir/fq2bam_output.bam \ --out-variants /outputdir/out.vcf \ --num-htvc-threads 8 \ --no-alt-contigs \#This flag will ignore all outputs after chrM --run-partition
```

Best Performance for minimap2

The following command line options provided optimal performance for PacBio data running on 2x 7742 + 2x A100 80GB PCIe.

```
$ # This command assumes all the inputs are in INPUT_DIR and all the outputs go to OUTPUT_DIR. docker run --rm --gpus all --volume INPUT_DIR:/workdir --volume OUTPUT_DIR:/outputdir \ --workdir /workdir --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun minimap2 \ --ref /workdir/${REFERENCE_FILE} \ --in-fq /workdir/${INPUT_FASTQ} \ --out-bam /outputdir/${OUTPUT_BAM} \ --num-chaining-threads 3 \ --alignment-large-pair-size 5000 \ --process-large-alignments-on-cpu \ --num-alignment-threads-per-gpu 8 \ --num-alignment-device-mem-buffers 8 \ --gpusort \ --gpuwrite
```

GDS Support

For additional performance improvements and final BAM writing bandwidth use GPUDirect Storage (GDS), part of the CUDA toolkit. Note that the system must be set up and supported to use GDS.

The following are references for setting up and using GDS:

- Overall NVIDIA GPUDirect Storage documentation
- NVIDIA GPUDirect Storage Installation and Troubleshooting Guide
- Using GDS in containers

```
# Using GDS with the convenience docker wrapper. $ wget https://raw.githubusercontent.com/NVIDIA/MagnumIO/main/gds/docker/gds-run-container $ chmod +x gds-run-container $ ./gds-run-container run \ --rm \ --gpus all \ --enable-mofed \ --enable-gds \ --volume INPUT_DIR:/workdir \ --volume OUTPUT_DIR:/outputdir \ --workdir /workdir \ --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun fq2bam \ --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-fq /workdir/fastq1.gz /workdir/fastq2.gz \ --out-bam /outputdir/fq2bam_output.bam \ --tmp-dir /workdir \ --out-recal-file recal.txt \ --knownSites /workdir/hg.known_indels.vcf \ --gpusort \ --gpuwrite \ --use-gds # Using GDS without the wrapper. $ docker run \ --ipc host \ --volume /run/udev:/run/udev:ro \ --device=/dev/nvidia-fs0 \ --device=/dev/nvidia-fs1 \ --device=/dev/nvidia-fs2 \ --device=/dev/nvidia-fs3 \ --device=/dev/nvidia-fs4 \ --device=/dev/nvidia-fs5 \ --device=/dev/nvidia-fs6 \ --device=/dev/nvidia-fs7 \ --device=/dev/nvidia-fs8 \ --device=/dev/nvidia-fs9 \ --device=/dev/nvidia-fs10 \ --device=/dev/nvidia-fs11 \ --device=/dev/nvidia-fs12 \ --device=/dev/nvidia-fs13 \ --device=/dev/nvidia-fs14 \ --device=/dev/nvidia-fs15 \ --rm \ --gpus all \ -enable-mofed \ --volume INPUT_DIR:/workdir \ --volume OUTPUT_DIR:/outputdir \ --workdir /workdir \ --env TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES=268435456 \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun fq2bam --ref /workdir/Homo_sapiens_assembly38.fasta \ --in-fq /workdir/fastq1.gz /workdir/fastq2.gz \ --out-bam /outputdir/fq2bam_output.bam \ --tmp-dir /workdir \ --out-recal-file recal.txt \ --knownSites /workdir/hg.known_indels.vcf \ --gpusort \ --gpuwrite \ --use-gds
```