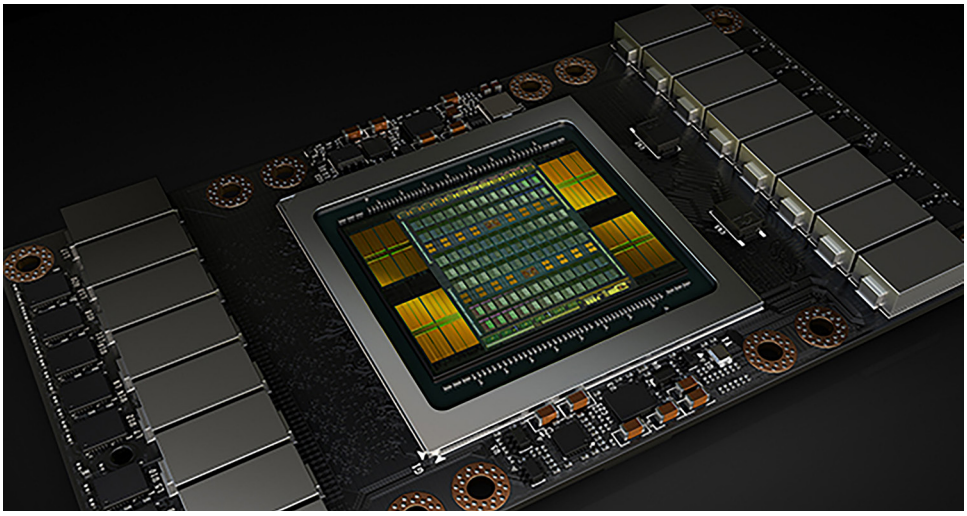
Running Calculations on GPUs with Gaussian 16

As a result of a fruitful, ongoing collaboration between Gaussian Inc., NVIDIA and Hewlett-Packard Enterprise, Gaussian 16 supports running calculations using NVIDIA GPUs. NVIDIA Tesla K40, Tesla K80, Tesla P100, Tesla V100 and A100 GPUs can be used in Hartree-Fock and DFT calculations, including energies, optimizations and frequencies, for ground and excited states (TD), and for closed shell and open shell molecules. ONIOM, SCRF solvation and all major properties are supported, as are all DFT functionals available in Gaussian 16, making the most frequently-run Gaussian calculations applicable to execution with GPUs.



NVIDIA A100 GPUs use the NVIDIA Ampere GPU architecture to achieve ~9.7 TFlops peak performance (double precision), and have 40GB HBM2 or 80GB HBM2e memory.

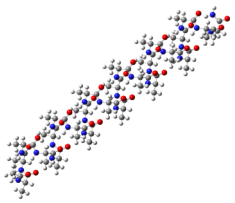
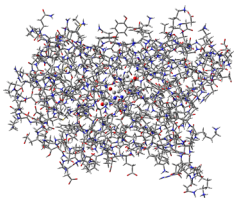
NVIDIA Tesla V100 GPUs *(illustrated above)* use the NVIDIA Volta GPU architecture to achieve ~7.8 (SXM2) or ~7 (PCIe) TFlops peak performance (double precision), and have 32GB and 16GB HBM2 memory (respectively).

NVIDIA Tesla P100 GPUs use the NVIDIA Pascal GPU architecture to achieve ~5 TFlops peak performance (double precision), and have 12-16GB HBM2 memory.

NVIDIA Tesla K40 & Tesla K80 GPUs have 12GB 5GHz GDDR5 VRAM and achieve peak performance of ~1.5 TFlops (double precision), with 1 and 2 GPUs per board (respectively).

Results for Example Calculations

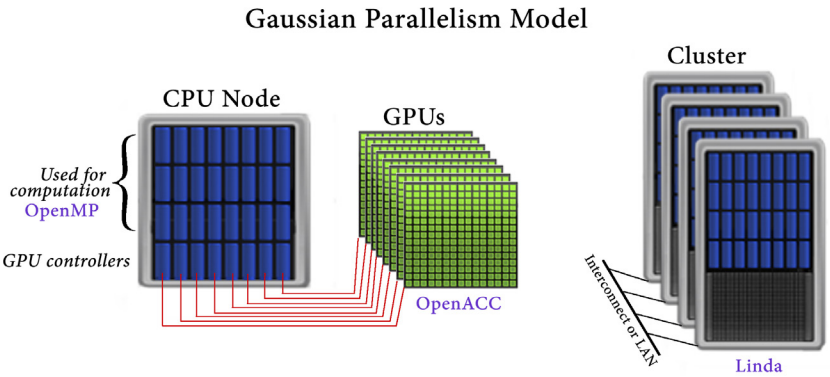
The following table provides some example performance data for the current version of GPU support. The timings compare running across 28 CPU cores on a DGX-1 server (Broadwell CPUs) versus running on the same server across 28 CPU cores and 8 GPUs (8 CPU cores serve as GPU controllers during the GPU-enabled parts of the calculation).

Molecule		Calculations	Speedup with GPUs
Alanine 25		APFD/6-31G(d) Freq	2.34
		TD APFD/6-31G(d) Freq	2.35
Green Fluorescent Protein (GFP)		ONIOM(APFD/6-311+G(2d,p):Amber)=Embed Freq	3.17
		TD ONIOM(APFD/6-311+G(2d,p):Amber)=Embed Freq	4.17

Server: dual Intel Xeon E5-2698 v4 CPUs (2.32GHz ; 20 cores/chip), 512GB 2133 MHz DDR4 RDIMM memory and 8 Tesla V100 SMX2 GPUs with 16GB HBM2 memory. Gaussian source code compiled with PGI Accelerator Compilers (18.10), CUDA (10.0) with OpenACC (2.5 standard).

Parallelization Strategy

Within Gaussian 16, GPUs are used for a small fraction of code that consumes a large fraction of the execution time. The implementation of GPU parallelism conforms to Gaussian's general parallelization strategy. Its main tenets are to avoid changing the underlying source code and to avoid modifications which negatively affect CPU performance. For these reasons, OpenACC was used for GPU parallelization



The Gaussian approach to parallelization relies on environment-specific parallelization frameworks and tools: OpenMP for shared-memory, Linda for cluster and network parallelization across discrete nodes, and OpenACC for GPUs.

The process of implementing GPU support involved many different aspects:

- Identifying places where GPUs could be beneficial. These are a subset of areas which are parallelized for other execution contexts because using GPUs requires fine grained parallelism.
- Understanding and optimizing data movement/storage at a high level to maximize GPU efficiency.

PGI's sophisticated profiling and performance evaluation tools were vital to the success of the effort.

NVIDIA HPC SDK: Compilers with OpenACC

The compilers included in the NVIDIA HPC SDK fully support the current OpenACC standard as well as important extensions to it. NVIDIA is a key contributor to the ongoing development of OpenACC.

OpenACC enables developers to implement GPU parallelism by adding compiler directives to their source code, often eliminating the need for rewriting or restructuring. For example, the following Fortran compiler directive identifies a loop which the compiler should parallelize:

```
!$acc parallel loop
```

Other directives allocate GPU memory, copy data to/from GPUs, specify data to remain on the GPU, combine or split loops and other code sections, and generally provide hints for optimal work distribution management, and more.

The OpenACC project is very active, and the specifications and tools are changing fairly rapidly. This has been true throughout the lifetime of this project. Indeed, one of its major challenges has been using OpenACC in the midst of its development. The talented people at PGI, now part of NVIDIA, were instrumental in addressing issues that arose in one of the very first uses of OpenACC for a large commercial software package.

Specifying GPUs to Gaussian 16

The GPU implementation in Gaussian 16 is sophisticated and complex, but using it is simple and straightforward. GPUs are specified with 1 additional Link 0 command (or equivalent *Default.Route* file entry/command line option). For example, the following commands tell Gaussian to run the calculation using 24 compute cores plus 8 GPUs+8 controlling cores (32 cores total):

`%CPU=0-31`	*Request 32 CPUs for the calculation: 24 cores for computation, and 8 cores to control GPUs (see below).*
`%GPUCPU=0-7=0-7`	*Use GPUs 0-7 with CPUs 0-7 as their controllers.*

Detailed information is available at **www.gaussian.com/gpu**.

Project Contributors

Roberto Gomperts
NVIDIA

Michael Frisch
Gaussian

Kyle Jacobs
NVIDIA

Brent Leback
NVIDIA/PGI

Giovanni Scalmani
Gaussian

Gaussian, Inc.
340 Quinnipiac St. Bldg. 40
Wallingford, CT 06492 USA
custserv@gaussian.com