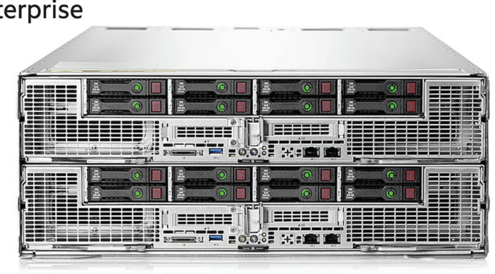
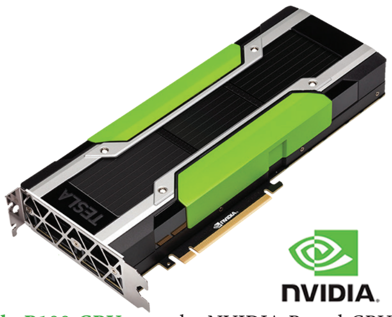
Running Calculations on GPUs with Gaussian 16

As a result of a fruitful, ongoing collaboration between the Gaussian Inc., NVIDIA and its PGI compiler team and Hewlett-Packard Enterprise, Gaussian 16 supports running calculations using NVIDIA GPUs. The first phase of this work is complete, and NVIDIA Tesla K40 and Tesla K80 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. The most frequently-run Gaussian calculations will be applicable to execution with GPUs.

Ongoing development is taking place on HPE Apollo 6500 systems containing NVIDIA Tesla P100 (Pascal) GPUs. The work takes advantage of PGI Accelerator compilers supporting OpenACC.



HPE Apollo 6500 Systems contain dual Intel Xeon E5-2600 v4 (Broadwell) processors and 4 or 8 NVIDIA Tesla P100 (Pascal) GPUs per CPU. They can be configured with SSD 12G SAS storage and 1024 GB DDR4 2400MHz memory.

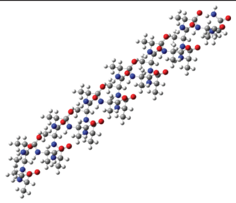
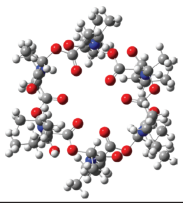
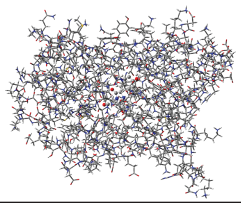


NVIDIA Tesla P100 GPUs use the NVIDIA Pascal GPU architecture to achieve ~5 TFlops peak performance (double precision), and have 12-16GB Chip-on-Wafer-on-Substrate HBM2 memory.

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 this initial version of GPU support. All calculations were run on a server system with 32 Haswell cores and 4 NVIDIA K80 2-GPU boards. The timings compare running on all of the 32 CPU cores alone vs. running on all 8 GPUs and the remaining available 24 CPU cores (with 8 cores used as GPU controllers).

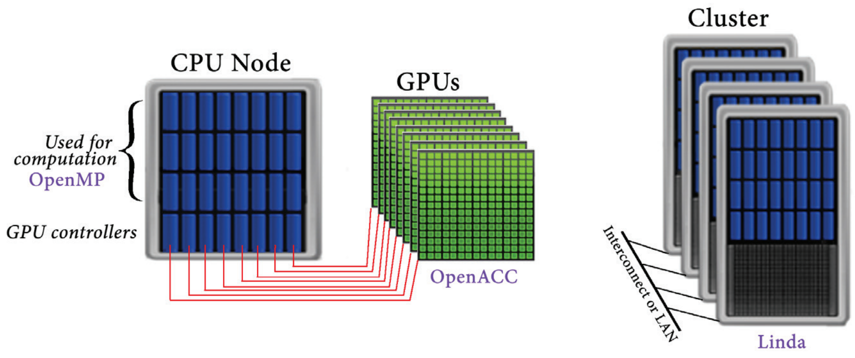
Molecule		Calculations	Speedup with GPUs
Alanine 25		APFD/6-31G(d) Freq	1.45
		TD APFD/6-31G(d) Freq	1.39
		APFD/6-311+G(2d,p) NMR SCRF	1.34
Valinomycin		wB97xD/6-311+(2d,p) Freq	2.25
GFP		ONIOM(APFD/6-311+G(2d,p):Amber)=Embed Freq	1.60
		TD ONIOM(APFD/6-311+G(2d,p):Amber)=Embed Freq	2.07

Server: dual Intel Xeon E5-2698 v3 CPUs (2.30GHz ; 16 cores/chip), 256GB memory and 4 Tesla K80 dual GPU boards (boost clocks: MEM 2505 and SM 875). Gaussian source code compiled with PGI Accelerator Compilers (16.5) 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.

Gaussian Parallelism Model



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.

PGI

PGI Accelerator Compilers with OpenACC

PGI compilers fully support the current OpenACC standard as well as important extensions to it. PGI is an important 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 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 on our website.

Project Contributors

Roberto Gomperts
NVIDIA

Michael Frisch
Gaussian

Brent Leback
NVIDIA/PGI

Giovanni Scalmani
Gaussian

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