

Running R & TMB with GPU

Gavin Fay

June 24, 2016

@kaskr was interested in running TMB using an Nvidia GPU using nvblas. Herein brief comments on my experience installing and testing the performance.

My system - I am running Ubuntu 14.04, and I have an Nvidia Quadro K2000 graphics card.

Installation

Brief summary of steps:

1. Install CUDA Toolkit

First ensure that you have a CUDA-capable GPU & download any necessary headers/etc.

I followed the pre-installation instructions at:

<http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#pre-installation-actions>

(in my case I already had the right dependencies)

Download the CUDA Toolkit (current version 7.5) for your system from

<https://developer.nvidia.com/cuda-downloads>

I used the local download installer (deb local). This can then be installed at command line, (e.g. by `sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb`).

Install CUDA by running

1. `$ sudo dpkg -i cuda-repo-ubuntu1404-7-5-local_7.5-18_amd64.deb`
2. `$ sudo apt-get update`
3. `$ sudo apt-get install cuda`

(Section 3 of the installation guide details package manager installation for other linux systems)

<http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#package-manager-installation>

The download link also provides instructions for other systems and installation options.

2. Modify PATH and LD_LIBRARY_PATH environment variables.

These are some post-installation instructions that need to be performed manually.

(necessary details below, but more info at <http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#post-installation-actions>)

```
$ export PATH=/usr/local/cuda-7.5/bin:$PATH
```

```
$ export LD_LIBRARY_PATH=/usr/local/cuda-7.5/lib:$LD_LIBRARY_PATH
```

You can verify the installation by following some instructions at

<http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#recommended-post>

(these are recommended but not necessary to get working)

3. Create a config file `nvblas.conf`

This is necessary.

There is an example on page 9 in the CUDA manual, that I copied to modify for my uses.

(https://www.ecse.rpi.edu/~wrf/wiki/ParallelComputingSpring2014/nvidia/cuda6doc/pdf/NVBLAS_Library.pdf)

The only thing I had to change in here was the `NVBLAS_CPU_BLAS_LIB` variable, that tells it what blas to use for CPU.

Mine is:

```
NVBLAS_CPU_BLAS_LIB /usr/lib/libblas.so
```

My `nvblas.conf` file is *here*.

In addition, I had to create the below variable to point to the config file.

4. Set a environment variable `NVBLAS_CONFIG_FILE` that points to the config file.

```
$ export NVBLAS_CONFIG_FILE=/path/to/config/file/nvblas.conf$NVBLAS_CONFIG_FILE
```

For example, I put my `nvblas.conf` file in my home directory, and so my `NVBLAS_CONFIG_FILE` variable was created by:

```
$ export NVBLAS_CONFIG_FILE=/home/gavin/nvblas.conf$NVBLAS_CONFIG_FILE
```

5. Identify Location of `libnvblas.so`

(this is so you can load this when opening R)

e.g. use `$ find / -name "libnvblas.so"`

On my machine this was installed at: `/usr/local/cuda-7.5/lib64/libnvblas.so`

6. Open R loading the `nvblas` library, invoking use of the GPU.

Enter

```
$ LD_PRELOAD=/path/to/libnvblas/libnvblas.so R
```

So given the results of step 5, I run R using:

```
$ LD_PRELOAD=/usr/local/cuda-7.5/lib64/libnvblas.so R
```

This will start an instance of R allowing access to the GPU.

7. Run R commands as normal.

Note that the approach above just uses R commands and functions as normal, no special R functions written specifically for optimization on the GPU are used.

Testing

To monitor GPU usage, in a separate shell I use `watch nvidia-smi`. In an additional shell I can monitor CPU usage using `top`.

Simple tests

Matrix multiplication & Cholesky decomposition I tried the following simple calculations on instances of R that either did or did not have the GPU loaded.

```
x <- diag(4e3)+1
system.time(x %*% x)
system.time(chol(x))
```

The shell on the left in the picture below shows the results with the GPU, the shell in the bottom-right is standard R.

Appreciable speed increases when calculations can be done on the GPU. The improvements in runtime increase as dimensions of problem increases.

gavin@DT-322203-SMAST:~\$ export NVBLAS_CONFIG_FILE=/home/gavin/nvblas.conf\$NVBLAS_CONFIG_FILE

gavin@DT-322203-SMAST:~\$ LD_PRELOAD=/usr/local/cuda-7.5/lib64/libnvblas.so R

[NVBLAS] Cannot Log File 'nvblas.log'

[NVBLAS] Using devices :0

[NVBLAS] Config parsed

[NVBLAS] Cannot Log File 'nvblas.log'

[NVBLAS] Using devices :0

[NVBLAS] Config parsed

[NVBLAS] Cannot Log File 'nvblas.log'

[NVBLAS] Using devices :0

[NVBLAS] Config parsed

R version 3.2.2 (2015-08-14) -- "Fire Safety"

Copyright (C) 2015 The R Foundation for Statistical Computing

Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

```
> x <- diag(4e3)+1
> system.time(x %*% x)
user system elapsed
3.571 1.092 4.660
>
> system.time(x %*% x)
user system elapsed
3.596 1.095 4.694
> system.time(x %*% x)
user system elapsed
3.623 1.080 4.704
> system.time(chol(x))
user system elapsed
1.287 0.518 1.803
> system.time(chol(x))
user system elapsed
1.269 0.514 1.791
> system.time(chol(x))
user system elapsed
1.199 0.509 1.709
>
```

Every 2.0s: nvidia-smi Fri Jun 24 17:43:43 2016

Fri Jun 24 17:43:43 2016

NVIDIA-SMI 352.63 Driver Version: 352.63									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC				
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.			
0	Quadro K2000	Off	0000:01:00.0	0n	N/A				
30%	33C	P8	N/A / N/A	132MiB / 2047MiB	0%	Default			

Processes:

GPU	PID	Type	Process name	GPU Memory Usage
0	3493	G	/usr/bin/X	96MiB
0	9234	C	/usr/lib/R/bin/exec/R	26MiB

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

```
> x <- diag(4e3)+1
> system.time(x %*% x)
user system elapsed
38.912 0.015 38.943
> system.time(x %*% x)
user system elapsed
38.971 0.004 38.991
> system.time(x %*% x)
user system elapsed
38.957 0.016 38.989
> system.time(chol(x))
user system elapsed
8.129 0.016 8.148
> system.time(chol(x))
user system elapsed
8.068 0.004 8.067
> system.time(chol(x))
user system elapsed
8.124 0.020 8.148
>
```

Testing TMB

The ar1_4D example Tested the ar1_4D example from the TMB examples folder.

```
system.time(source("ar1_4D.R"))
```

I ran the ar1_4D model with both n=8 and n=10.

For n=8, improvement with GPU was minor.

For n=10, with GPU was >2x faster than when run without GPU.

scenario	n	user	system	elapsed
with nvblas	8	50.327	3.250	53.608
normal R	8	60.794	0.105	60.928
with nvblas	10	240.180	24.160	264.414
normal R	10	578.301	0.681	579.208