GPU computing with the gputools package

April 12, 2010

Overview

- Introduction.

Overview

- Introduction.
- The package.

Overview

- Introduction.
- The package.
- W(h)ither gputools?

Overview

- Introduction.
- The package.
- W(h)ither gputools?
- Audience participation.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.
- Highly parallel hardware with 32–bit vector–processing capabilities.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.
- Highly parallel hardware with 32–bit vector–processing capabilities.
- Early numerical applications appear to be due to physicists (cf. www.gpgpu.org):
  - Lattice–Boltzmann computations: Li et al., 2002.
  - Boundary–value problems: Goodnight et al., 2003.
  - Matrix algebra, dynamic applications: Moravanszky, 2003.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.
- Highly parallel hardware with 32–bit vector–processing capabilities.
- Early numerical applications appear to be due to physicists (cf. www.gpgpu.org):
  - Lattice–Boltzmann computations: Li et al., 2002.
  - Boundary–value problems: Goodnight et al., 2003.
  - Matrix algebra, dynamic applications: Moravanszky, 2003.
- Require specialized drivers, software to program - not easy.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.
- Highly parallel hardware with 32–bit vector–processing capabilities.
- Early numerical applications appear to be due to physicists (cf. www.gpgpu.org):
  - Lattice–Boltzmann computations: Li et al., 2002.
  - Boundary–value problems: Goodnight et al., 2003.
  - Matrix algebra, dynamic applications: Moravanszky, 2003.
- Require specialized drivers, software to program - not easy.
- API's from NVidia ("CUDA") and ATI/AMD now freely available.

GPU ≡ graphical processing unit

- Special–purpose coprocessor for graphics applications.
- Highly parallel hardware with 32–bit vector–processing capabilities.
- Early numerical applications appear to be due to physicists (cf. www.gpgpu.org):
  - Lattice–Boltzmann computations: Li et al., 2002.
  - Boundary–value problems: Goodnight et al., 2003.
  - Matrix algebra, dynamic applications: Moravanszky, 2003.
- Require specialized drivers, software to program - not easy.
- API's from NVidia ("CUDA") and ATI/AMD now freely available.
- Math–capable GPU's are now inexpensive. Standard equipment on many computers, including laptops.

- GPU–enabled numerical software becoming available commercially.
  - Jacket, a Matlab accessory from Acclereyes.
  - Mathematica support.
  - Numerous standalone packages on NVidia website.

- GPU–enabled numerical software becoming available commercially.
  - Jacket, a Matlab accessory from Acclereyes.
  - Mathematica support.
  - Numerous standalone packages on NVidia website.
- Why not R?

- GPU–enabled numerical software becoming available commercially.
  - Jacket, a Matlab accessory from Acclereyes.
  - Mathematica support.
  - Numerous standalone packages on NVidia website.
- Why not R?
- Buckner et al. release "gputools" 0.1 in spring, 2009.
  - Tools related to that group's work looking for causal relations in gene–expression data.

- GPU–enabled numerical software becoming available commercially.
  - Jacket, a Matlab accessory from Acclereyes.
  - Mathematica support.
  - Numerous standalone packages on NVidia website.
- Why not R?
- Buckner et al. release "gputools" 0.1 in spring, 2009.
  - Tools related to that group's work looking for causal relations in gene–expression data.
- Co–collaboration leads to paper submitted last summer.
  - "The gputools package enables GPU computing in R"
  - Buckner, Wilson, Seligman, Athey, Watson, Meng
  - *Bioinformatics,* 2010 26(1):134–135

- GPU–enabled numerical software becoming available commercially.
  - Jacket, a Matlab accessory from Acclereyes.
  - Mathematica support.
  - Numerous standalone packages on NVidia website.
- Why not R?
- Buckner et al. release "gputools" 0.1 in spring, 2009.
  - Tools related to that group's work looking for causal relations in gene–expression data.
- Co–collaboration leads to paper submitted last summer.
  - "The gputools package enables GPU computing in R"
  - Buckner, Wilson, Seligman, Athey, Watson, Meng
  - *Bioinformatics,* 2010 26(1):134–135
- Remains very much a work in progress.

Package highlights

- Contributions from MBNI team include:
  - Correlation - Pearson and Kendall (JB/JW): **cor()**
  - Granger causality (JB): **granger.test** from **MSBVAR**
  - Hierarchical clustering (JB/JW): **hclust**
  - Spline–based mutual information (JB)
  - Matrix multiplication (cudablas wrapper): **%*%**
  - SVM training (wrapper): **svm** from **e1071**
  - SVD (wrapper): **fastICA** package
  - attendant functions and package layout

- Contributions from MLS include:
  - Linear, generalized linear modeling: **lm(), glm()**
  - Least–squares fit: **lsfit()**
  - Rank–revealing QR decomposition: **qr()**
  - Blocked, partial–pivoting QR
  - Matrix cross–products: **crossprod()**

Differences in contribution reflect complementary approaches

- JB:
  - Higher–level, although some key lower–level functions (e.g., matrix multiplication) also implemented.
  - Less oriented toward traditional numerical linear algebra, so able to exploit richer set of concurrent algorithms.
  - By same token, implementation relies less on lower–level libraries and more on hand–coded parallelism.
  - Relatively small communication costs result in 10x – 50+x speedup over CPU–only implementations.

- MLS:
  - Mostly lower–level utiltities, with same look and feel as their base–package counterparts.
  - More like traditional NLA. In fact QR decomposition drives much of the work.
  - Relies much more heavily on low–level libraries, viz., **cudaBlas**.
  - Communication costs higher (think Householder transformations and block updates). 1000×1000 matrix needed for breakeven, more like 4000×4000 needed to start seeing 10x. On the bright side, though, much bigger matrices now treatable in "user time".

Hardware, tools requirements

- CUDA–supporting GPU. Radeon not supported yet.

Hardware, tools requirements

- CUDA–supporting GPU. Radeon not supported yet.
- CUDA driver and development tools, available as free downloads from NVidia: compiler, libraries (cudaBLAS).

Hardware, tools requirements

- CUDA–supporting GPU. Radeon not supported yet.
- CUDA driver and development tools, available as free downloads from NVidia: compiler, libraries (cudaBLAS).
- gputools v0.2 supported on Linux, 32–bit Mac; available from CRAN.

Beta versions

Downloadable from:
http://brainarray.mbni.med.umich.edu/brainarray/rgpgpu/

Hardware, tools requirements

- CUDA–supporting GPU. Radeon not supported yet.
- CUDA driver and development tools, available as free downloads from NVidia: compiler, libraries (cudaBLAS).
- gputools v0.2 supported on Linux, 32–bit Mac; available from CRAN.

Beta versions

Downloadable from:
http://brainarray.mbni.med.umich.edu/brainarray/rgpgpu/

- Just download and install. Run–time environment checks for presence of the GPU. Emulator runs if no GPU present.

Hardware, tools requirements

- CUDA–supporting GPU. Radeon not supported yet.
- CUDA driver and development tools, available as free downloads from NVidia: compiler, libraries (cudaBLAS).
- gputools v0.2 supported on Linux, 32–bit Mac; available from CRAN.

Beta versions

Downloadable from:
http://brainarray.mbni.med.umich.edu/brainarray/rgpgpu/

- Just download and install. Run–time environment checks for presence of the GPU. Emulator runs if no GPU present.
- Familiar **R** commands prefaced by "gpu". E.g., gpuLm(), gpuCor(), ...

Some conclusions

- NLA–style kernels make heavy use of **cudaBLAS** calls. Very little device–level programming required for these. Key concerns here are minimizing communication, exploiting data locality - e.g., blocking.

Some conclusions

- NLA–style kernels make heavy use of **cudaBLAS** calls. Very little device–level programming required for these. Key concerns here are minimizing communication, exploiting data locality - e.g., blocking.
- These types of kernels have large breakeven sizes. The 1000×1000 observed for QR is in line with the literature, however.

Some conclusions

- NLA–style kernels make heavy use of **cudaBLAS** calls. Very little device–level programming required for these. Key concerns here are minimizing communication, exploiting data locality - e.g., blocking.
- These types of kernels have large breakeven sizes. The 1000×1000 observed for QR is in line with the literature, however.
- Some utilities exhibit more concurrency and achieve more dramatic speedups, with much lower breakeven size. These tend to require more device–level implementation, however. These tend to be less like kernels and more like higher–level applications.

- Single–precision seems to be "good enough" for one–off invocations, but where does this begin to break down?

- Single–precision seems to be “good enough” for one–off invocations, but where does this begin to break down?

- Questions:

  Are current problems of interest large enough to benefit from these speedups? Will we need to “expose the kernel”?

- Single–precision seems to be "good enough" for one–off invocations, but where does this begin to break down?

- Questions:

  Are current problems of interest large enough to benefit from these speedups? Will we need to "expose the kernel"?

  - Tracking R–base is a software–engineering hassle.

Low–hanging fruit

- Double precision
  - DP now available in low–priced boards
  - SP / DP ratio moving from 8x to 2x

Low–hanging fruit

- Double precision
  - DP now available in low–priced boards
  - SP / DP ratio moving from 8x to 2x
- Rank–revealing options, as applicable.

Low–hanging fruit

- Double precision
  - DP now available in low–priced boards
  - SP / DP ratio moving from 8x to 2x
- Rank–revealing options, as applicable.
- Sampling, resampling tools
  - CUDA–ready Mersenne Twister
  - rnorm(), rgamma(), ...
  - Ready applications in bootstrapping, MCMC
  - Also a CUDA–ready QRNG.

Low–hanging fruit

- Double precision
  - DP now available in low–priced boards
  - SP / DP ratio moving from 8x to 2x
- Rank–revealing options, as applicable.
- Sampling, resampling tools
  - CUDA–ready Mersenne Twister
  - rnorm(), rgamma(), ...
  - Ready applications in bootstrapping, MCMC
  - Also a CUDA–ready QRNG.
- Benchmarking: vs. **mkl**, as well as tuned libraries. Especially, identifying the break–even points

Low–hanging fruit

- Double precision
  - DP now available in low–priced boards
  - SP / DP ratio moving from 8x to 2x
- Rank–revealing options, as applicable.
- Sampling, resampling tools
  - CUDA–ready Mersenne Twister
  - rnorm(), rgamma(), ...
  - Ready applications in bootstrapping, MCMC
  - Also a CUDA–ready QRNG.
- Benchmarking: vs. **mkl**, as well as tuned libraries. Especially, identifying the break–even points
- Integration with other packages.

Medium term

- AMD/ATI Radeon support, openCL

Medium term

- AMD/ATI Radeon support, openCL
- Mutlicore–aware implementations

Medium term

- AMD/ATI Radeon support, openCL
- Mutlicore–aware implementations
- Exposing kernels

Long term

- Playing well in all sorts of hardware environment: multiple GPU, CPU; clusters; clouds ...

Long term

- Playing well in all sorts of hardware environment: multiple GPU, CPU; clusters; clouds ...
- Seamless integration: hardware details under the covers. Do you really want to preface everything with "gpu"?

Long term

- Playing well in all sorts of hardware environment: multiple GPU, CPU; clusters; clouds ...
- Seamless integration: hardware details under the covers. Do you really want to preface everything with "gpu"?
- Assimilation.

Acknowledgments

The MBNI group wish to acknowledge their funding sources:

- J. Buckner, J. Wilson, and F. Meng are members of the Pritzker Neuropsychiatric Disorders Research Consortium, which is supported by the Pritzker Neuropsychiatric Disorders Research Fund LLC.
- This work is also partly supported by the National Center for Integrated Biomedical Informatics through NIH grant 1U54DA021519-01A1 to the University of Michigan.

MLS wishes to thank:

- Roger Ngouenet, XL Solutions
- Mark Troll, Rapid Biologics and Aaaron Thermal Technologies
- Dirk Eddelbuettel
- Chris Fraley, Insilicos LLC and UW Statistics
- Anne Greenbaum, UW AMath