GPU-PARALLELIZED LEVENBERG-MARQUARDT MODEL FITTING TOWARDS REAL-TIME AUTOMATED PARAMETRIC DIFFUSION NMR IMAGING

Marco Palombo¹,²

D. Zhang³, X. Zhu⁴, J. Valette¹, A. Gozzi⁵, A. Bifone⁵, A. Messina⁶, G. Lamanna⁷, S. Capuani²

¹MIRCen, FBM, DSV, CEA Fontenay-aux-Roses, France
²IPCF-UOS Roma, Physics Dpt., Sapienza University, Rome, Italy
³ITG, Beckman Institute for Advanced Science & Technology, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA
⁴College of Economics & Management, China Agricultural University, Beijing, China
⁵IIT Center for Neuroscience and Cognitive Systems @ UniTn, Rovereto, Italy
⁶Physics Dpt. Sapienza University, Rome, Italy
⁷National Institute of Nuclear Physics, Pisa Section, Pisa, Italy

12 SEPTEMBER 2014
In this contribution we report one of the main goals of the GPU Application Project (GAP), which aims to investigate the deployment of Graphic Processing Units (GPU) in different context of realtime scientific applications.

In particular we focused on the application of GPUs in reconstruction of diffusion weighted nuclear magnetic resonance (DW-NMR) images by using non-Gaussian diffusion models.
OUTLINE

- State of the art of non-Gaussian DW-NMR imaging
- Why and how using GPU to make DW-NMR imaging faster
- Application: non-Gaussian DW-NMR imaging of mouse brain on GPU
- Future perspectives about GPU applications in NMR
- Conclusion
Typical DW-NMR imaging post-processing

\[ \text{Signal}(k, t) \propto \int P(r, t) e^{-ik \cdot r} dr \]

Weight of the NMR signal in diffusion (\textit{directional})

\[ b = k^2 t \hat{k} \]

In \textit{homogeneous} media \( P(r, t) \) is \textbf{Gaussian}

In \textit{heterogeneous} complex media \( P(r, t) \) is \textbf{not Gaussian}
128x128 pixel image x number of slice (~10-40) x number of diffusion gradient directions (≥15)
Typical DW-NMR imaging post-processing

128x128 pixel image x number of slice (~10-40) x number of diffusion gradient directions (≥15)
Typical non-Gaussian DW-NMR imaging post-processing

- Cumulant Expansion approach: Diffusional Kurtosis Imaging
- Stretched Exponential model Imaging
Why using non-Gaussian DW-NMR imaging post-processing

Nasopharyngeal Carcinoma

Maps of Gaussian and non-Gaussian parameters for the primary tumor

Yuan, Jing, et al. PloS one 2014 9(1); e87024.
Why using non-Gaussian DW-NMR imaging post-processing

Nasopharyngeal Carcinoma

Maps of Gaussian and non-Gaussian parameters for the metastatic node

Yuan, Jing, et al. PloS one 2014 9(1); e87024.
Why using non-Gaussian DW-NMR imaging post-processing

Ischemic Stroke

Non Gaussian diffusion maps best representing the ischemic lesion in each of the three animals.

Non-Gaussian parameters best represent the layered structure in the lesions and WM tracts.

Why using non-Gaussian DW-NMR imaging post-processing

Non-Gaussian DW-NMR imaging increases the specificity and sensitivity of DW-NMR techniques in determining pathological tissue (in both diagnosis and surgery phases)

Limitations:
The non-Gaussian processing is not currently available in “real-time” mode (in a few seconds)
$$S(b) \propto e^{-D_{app} b + \frac{1}{6} K_{app} [D_{app} b]^2 + \ldots}$$

$$S(b) = S(0) \exp \left[ -b \sum_{i,j} n_i n_j D_{ij} + \frac{1}{6} b^2 \left( \frac{1}{3} \sum_i D_{ii} \right)^2 \sum_{i,j,k,l} n_i n_j n_k n_l W_{ijkl} + O(b^3) \right]$$

N ≥ 15 independent measures to get $W_{ijkl}$

$$K(n) = \frac{\overline{D^2}}{[D(n)]^2} \sum_{i,j,k,l=1}^3 n_i n_j n_k n_l W_{ijkl}.$$
Conventional DTI
\[128 \times 128 \times 32 \times 15\]
\[= 7.864320 \times 10^6\]
linear systems to solve
\[\sim 15 \text{ s on CPU}^*\]

Kurtosis tensor imaging
\[128 \times 128 \times 32 \times 15\]
\[= 7.864320 \times 10^6\]
non-linear fit to perform
\[\sim 7200 \text{ s on CPU}^*\]

* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz
Conventional DTI
$128\times128\times32\times15$
$= 7.864320 \times 10^6$
linear systems to solve
$\sim 15 \text{ s on CPU}^*$

Kurtosis tensor imaging
$128\times128\times32\times15$
$= 7.864320 \times 10^6$
non-linear fit to perform
$\sim 7200 \text{ s on CPU}^*$

* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz
\[ S(b) = S(0)e^{-A_1 \left(b_1^* \right)^{\gamma_1}} - A_2 \left(b_2^* \right)^{\gamma_2} - A_3 \left(b_3^* \right)^{\gamma_3} \]

\[ b_i^* = \vec{b} \cdot \vec{\eta}_i \]

Anomalous diffusion reference frame

DTI reference frame

Anomalous Diffusion reference frame

Pisa | 12 September 2014
Conventional DTI
128x128x32x15
= 7.864320 \times 10^6
linear systems to solve
\sim 15 \text{ s on CPU*}

Stretched exponential imaging
128x128x32x15
= 7.864320 \times 10^6
non-linear fit to perform
\sim 6600 \text{ s on CPU*}

* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz
Conventional DTI
128x128x32x15
= 7.864320 \times 10^6
linear systems to solve
\sim 15 \text{ s on CPU}^*  

Stretched exponential imaging
128x128x32x15
= 7.864320 \times 10^6
non-linear fit to perform
\sim 6600 \text{ s on CPU}^*

* 8 threads on an Intel Xeon E5-2609 CPU at 2.4 GHz
• **Computationally intensive** — The time spent on computation significantly exceeds the time spent on transferring data to and from GPU memory.

• **Massively parallel** — The computations can be broken down into hundreds or thousands of independent units of work.
**Computationally intensive**

**Kurtosis tensor imaging**
- non-linear fit
- $\sim 7$ ms per pixel
- $\sim 10^{-80}$ ns to read data from global memory

**Stretched exponential imaging**
- non-linear fit
- $\sim 6$ ms per pixel
- $\sim 10^{-80}$ ns to read data from global memory

**Massively parallel**

**Kurtosis tensor imaging**
- $\sim 10^6$-$10^7$ independent non-linear fit

**Stretched exponential imaging**
- $10^6$-$10^7$ independent non-linear fit
- The Levenberg-Marquardt algorithm is based on an iterative numerical optimization procedure that minimizes the sum of squared model residuals.

- The CUDA kernel performs the Levenberg-Marquardt algorithm. This kernel maps each CUDA thread to a voxel, and it launches as many threads as voxels contained in a particular slice [1,2].

- Because processing of different voxels is totally independent, the threads do not need to synchronize.


WHY AND HOW USING GPU IN DW-NMR IMAGING

Model function: $f(x_i, \xi)$

Experimental data: $y_i$

Tentative result: $P_i = \{\xi\}_i$

Initial parameters $\{\xi\}_i$

Compute the jacobian: $J = \frac{\delta f}{\delta \xi}$

Compute the cross-product jacobian: $H = J^T J$

Compute the error gradient: $G = J^T (y_i - f(x_i, \xi))$

Solve $(H + \lambda \text{diag}(H))\mu = G$

Obtain $\mu$

\[ \xi = \xi + \mu \]

STOP and take $\xi$

Satisfy convergence criteria

\[ \langle (y_i - f(x_i, \xi))^2 \rangle \]

If $\langle (y_i - f(x_i, \xi))^2 \rangle \uparrow$

\[ \lambda \uparrow \]

Reject

If $\langle (y_i - f(x_i, \xi))^2 \rangle \downarrow$

\[ \lambda \downarrow \]

Accept

Yes

No
WHY AND HOW USING GPU IN DW-NMR IMAGING

GPU device: A NVIDIA GPU device can have a number of multiprocessors, each of which executes in parallel with the others.

Multiprocessor (MP):

MP:
Multiple CUDA blocks (and thus multiple GPU-LMFit fittings) can execute concurrently on one MP.

CUDA block:
One GPU-LMFit completes one LM fitting in one CUDA block with many parallel CUDA threads.

Host memory

Global memory

GPU optimized Levenberg-Marquardt algorithm for non-linear fitting.

A Nvidia Quadro K2000 with 2Gb of dedicated memory, which supports 1024 threads per block, with a maximum number of 64 registers per thread, was used for the analysis.

<table>
<thead>
<tr>
<th>DW-NMR data-set</th>
<th>≈ 121.521 Mb ≈ 5x10^5 voxels</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA blocks</td>
<td>8</td>
</tr>
<tr>
<td>CUDA Grid</td>
<td>(4096, 1, 1)</td>
</tr>
<tr>
<td>Total used global memory on GPU</td>
<td>≈ 7 – 15 Mb</td>
</tr>
<tr>
<td>Average Speed</td>
<td>≈ 12000 fit/s GPU ≈ 50 fit/s CPU</td>
</tr>
<tr>
<td>Computing time</td>
<td>≈ 40 s GPU ≈ 9720 s CPU</td>
</tr>
<tr>
<td>Speed-up factor</td>
<td>≈ 240</td>
</tr>
</tbody>
</table>
GPU optimized Parallel Tempering algorithm for non-linear fitting.

A Nvidia Quadro K2000 with 2Gb of dedicated memory, which supports 1024 threads per block, with a maximum number of 64 registers per thread, was used for the analysis.

<table>
<thead>
<tr>
<th></th>
<th>DW-NMR data-set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≈ 121.521 Mb</td>
</tr>
<tr>
<td></td>
<td>≈ 5×10^5 voxels</td>
</tr>
<tr>
<td>CUDA blocks</td>
<td>8</td>
</tr>
<tr>
<td>CUDA Grid</td>
<td>(4096, 1, 1)</td>
</tr>
<tr>
<td>Total used global memory on GPU</td>
<td>≈ 7 – 15 Mb</td>
</tr>
<tr>
<td>Average Speed</td>
<td>≈ 18000 fit/s GPU</td>
</tr>
<tr>
<td></td>
<td>≈ 50 fit/s CPU</td>
</tr>
<tr>
<td>Computing time</td>
<td>≈ 30 s GPU</td>
</tr>
<tr>
<td></td>
<td>≈ 9720 s CPU</td>
</tr>
<tr>
<td>Speed-up factor</td>
<td>≈ 360</td>
</tr>
</tbody>
</table>
Future Perspectives: DW-NMR Imaging

Potentiality of GPU optimized algorithms for non-linear fitting.

Simulated expected speed-up for non-Gaussian DW-NMR Imaging from commercial and one of the best Nvidia GPUs available

Potential speed-up thanks to one of the best Nvidia GPU available

<table>
<thead>
<tr>
<th>DW-NMR data-set</th>
<th>(\approx 121.521 \text{ Mb} ) (\approx 5 \times 10^5 \text{ voxels} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA blocks</td>
<td>8</td>
</tr>
<tr>
<td>CUDA Grid</td>
<td>((4096, 1, 1))</td>
</tr>
<tr>
<td>Total used global memory on GPU</td>
<td>(\approx 7 \text{ – 15 Mb} )</td>
</tr>
<tr>
<td>Average Speed</td>
<td>(\approx \text{ fit/s GPU} ) (\approx 50 \text{ fit/s CPU} )</td>
</tr>
<tr>
<td>Computing time</td>
<td>(\approx 10 \text{ s GPU} ) (\approx 9720 \text{ s CPU} )</td>
</tr>
<tr>
<td>Speed-up factor</td>
<td>(\approx 1000 )</td>
</tr>
</tbody>
</table>

Pisa | 12 September 2014
CONCLUSION

- A pixel-wise approach by using fast, accurate and robust parallel minimization optimizers, was implemented on GPU.

- A **dramatic speed-up** (~$10^2$-$10^3$) in massive non-linear model fitting analysis was obtained with respect to new generation multi-core processors.

- These results suggest that **real-time** automated pixel-wise parametric DW-NMR imaging is a promising application of GPUs.
Thank you for your attention

Special thanks to my co-authors

If interested, please contact me at:
marco.palombo@cea.fr