GPU VS CPU CLUSTER RECONSTRUCTION FOR LOW LATENCY ITERATIVE RECONSTRUCTION OF FIRST PASS STRESS CARDIAC PERFUSION WITH PHYSIOLOGICAL STRESS

Sébastien Roujol[1], Tamer A Basha[1], Christophe Schülke[1,2], Martin Buehrer[1,2], Warren J Manning[1,3], and Reza Nezafat[1]
[1]*Medecine, BIDMC / Harvard Medical School, Boston, MA, United States,* [2]*Institute for Biomedical Engineering, University and ETH Zurich, Zurich, Switzerland,* [3]*Radiology, BIDMC / Harvard Medical School, Boston, MA, United States*

Introduction: Exercise stress is the most commonly used stress protocol in imaging. 60-70% of nuclear stress and over 80% of our stress echocardiographic studies are performed during physical exercise, thereby providing physiologic and hemodynamic data. However in CMR, the vast majority of CMR perfusion studies are performed using pharmacological stress. We have recently installed an MR-compatible supine bicycle mounted on the scanner table which allows performing CMR perfusion immediately after physiologic stress. While in pharmacological stress, patients are able to hold their breath, in post-physical stress patients are unable to sustain a breath hold after physical exercise, limiting the choice of acceleration techniques such as k-t approaches [1] for perfusion. Additionally, due to subject motion during exercise, coil sensitivity map are inaccurate resulting in imaging artifacts in conventional parallel imaging reconstruction. Compressed sensing (CS) [2] is an alternative acceleration technique that enables high acceleration even without exploiting temporal dimension or need for coil maps. However, iterative CS reconstruction of randomly undersampled k-space is lengthy, performed off-line and is not usually integrated into the workflow of a clinical scan requiring viewing and initial assessment on the scanner console and storing the clinical images on the hospital PACS system. In this proposal, a graphic processing unit (GPU)-based workflow and a cluster-based workflow have been developed and compared to accelerate the iterative CS reconstruction and minimize the overall reconstruction latency.

Methods: Figure 1 shows the two workflows of the accelerated CMR perfusion reconstruction. After completion of the prospective CS undersampled CMR perfusion sequence, the reconstruction process is manually started by CMR technologist using an in-house graphical user interface developed in Matlab (The MathWorks, Natick, MA). All the subsequent reconstruction steps are then performed automatically without any user interaction. The raw data are pre-processed on the scanner workstation and sent to an external system for reconstruction and finally sent back to the scanner workstation and the PACS database. Pre- and post-processing are performed using the ReconFrame platform (Gyrotools, Zurich, Switzerland). The CS reconstruction is based on the fast alternating minimization approach [3] and is offloaded to an external system to enable high performance computing. Two approaches were investigated for high performance computing using either a dedicated workstation equipped with a GPU (Intel Xeon X5550, 2.66 GHz, 4Gb of RAM, GPU card : NVIDIA Tesla C2075) or using the shared Orchestra high performance compute cluster at Harvard Medical School (5500+ compute cores, >40TB of memory). A GPU-based implementation and a cluster based implementation of the CS reconstruction with total variation regularization based on the fast alternating minimization approach [3] were developed. Since this reconstruction is iterative and voxel-independent for each iteration, the parallelization level of the GPU implementation was set to the voxel level. Since this parallelization strategy would introduce intensive communication between computers via the network communication interface and substantial latency, the parallelization level of the cluster-based implementation was set to the image level. For comparison, a conventional CPU-based reconstruction (non-parallelized implementation) employing the CS reconstruction code provided in [3] was also tested and performed on the dedicated GPU workstation. The presented workflows have been tested in healthy subjects using a 1.5T Philips scanner and a prospective 4× CS-accelerated CMR perfusion sequence. Time and latency of each reconstruction step are reported.

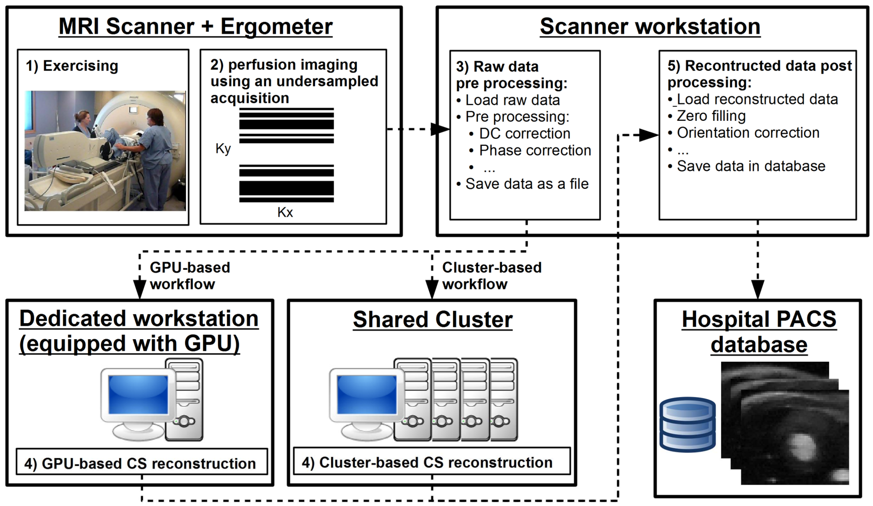

Figure 1. Workflows of the iterative CS reconstruction for CMR physiological stress perfusion using either a dedicated GPU workstation or a shared computer cluster.

	CPU-based workflow	GPU-based workflow	Cluster-based workflow
User interaction	10s	10s	10s
Raw data pre-processing	50s	50s	50s
Send data to external system	14s	14s	3 min, 30s
Thread/job creation	-	-	1 min, 50s
Reconstruction	51 min	4 min	5 to 30 min
Send data to scanner workstation	1s	1s	5s
Reconstructed data post-processing	5s	5s	5s
Overall reconstruction latency	**52 min, 20s**	**5 min, 25s**	**> 11 min, 30s**

Table 1. Time of the different reconstruction steps obtained for the treatment of one complete perfusion datasets (32 channels, 90 dynamics, 3 slices, image size=132×132, and 100 iterations for the reconstruction). The proposed GPU-based workflow provides a highly accelerated reconstruction with a guaranteed minimal latency.

Results: The GPU-based workflow allows reconstruction and viewing of the CS accelerated perfusion on the scanner console with a minimal guaranteed latency of 5 min 15s for the reconstruction of 3 slices and 90 dynamics (~10× faster than CPU-based workflow). The shared cluster-based workflow provided an overall reconstruction latency >11.5 min and is limited by long data transfer time, significant overhead associated to the job creation, and unpredictable waiting time for available resources.

Conclusions: A dedicated GPU based CS-reconstruction significantly improved the reconstruction time and guarantee a minimal latency required for optimized clinical MR protocol for CMR perfusion during physical stress. Although a dedicated CPU cluster may mitigate the latency, it is also associated with substantial increased cost compared to affordable (~300-2000$) GPU hardware.

References: [1] Tsao, MRM, 2003 [2] Lustig, MRM, 2007 [3]Yang,IEEE JSTSP,2010