A Generic, Multi-Node, Multi-GPU Reconstruction Framework for Online, Real-Time, Low-Latency MRI

Haris Saybasili[1], Daniel A. Herzka[2], Kestutis Barkauskas[3], Nicole Seiberlich[3], and Mark A. Griswold[1]
[1]Radiology, Case Western Reserve University, Cleveland, OH, United States, [2]Biomedical Engineering, Johns Hopkins University School of Medicine, Baltimore, MD, United States, [3]Biomedical Engineering, Case Western Reserve University, Cleveland, OH, United States

TARGET AUDIENCE: Computer Scientists and Engineers working in MRI; people interested in low-latency image reconstruction involving complex strategies.

PURPOSE: In MRI research, customizable, flexible external reconstruction frameworks are of great importance. Due to limited flexibility and freedom of the development environments provided by MRI scanner vendors, many research sites use self-developed external reconstruction frameworks to test and deploy their new imaging methods and ideas. Various image reconstruction environments have been presented in recent years [1,2].Even though these environments are flexible, they do not support distributed environment, such as multi-node, or multi-GPU systems in a transparent manner. Additionally, they may require several high-level software libraries that will negatively affect code portability, which complicates debugging, development, and maintenance. In this work, we present an automatically distributed, highly flexible external reconstruction environment developed using only low-level system libraries (thus, lightweight) for improved code portability and decreased code complexity. Our framework is capable of transparently distributing reconstruction tasks across workstations connected via network (nodes). In addition to remote distribution, each node (workstation) detects available GPUs, and locally distributes its reconstruction task for improved performance.

METHODS: ***Software Implementation**:* Image reconstruction was modeled as a pipeline, or series of *services,* where each *service* forwards its output to the next *service* in chain. A generic pipeline manager object was provided to automatically configure and manage reconstruction pipelines. An extra execution thread (optional) was transparently provided to each *service* by the framework to enable asynchronous computations that may be required during dynamic imaging (e.g. TGRAPPA calibration). Each *service* had the capacity to perform coil selective (subset of all the coils) reconstructions to enable distributed execution (both locally on multiple GPUs and remotely on multiple nodes). Local and remote task distribution was totally transparent to the user. One node was assigned as the *master* node to distribute the tasks to the reconstruction nodes. The list of reconstruction nodes was provided as a command line parameter to the runtime system on the *master* node for remote task distribution. The pipeline manager on each node automatically detected the number of local GPUs, and the reconstruction tasks were automatically distributed to each GPU to be executed on a separate thread. Each GPU executed the same pipeline, on different portions of the raw MRI data. Final image was calculated on the *master* node from all partial results. Figure 1 depicts the distributed reconstruction process on N nodes with M GPUs, for a reconstruction pipeline of *K services*. ***Tools:*** C++ was used as the programming language. CUDA 4.0 (http:// http://developer.nvidia.com/cuda-toolkit-40) was used for GPU programming. POSIX threads were used for multi-threading. A real-time product sequence was modified to configure the reconstruction pipeline from the sequence user interface. An additional module was implemented on the scanner to communicate with our reconstruction environment. TCP/IP protocol was used for network programming. ***Hardware:*** Each reconstruction node had an Intel Xeon X5660 CPU (6 cores, 12 threads), 48 GB of RAM and two NVIDIA Fermi M2090 GPUs. Inter-node communications were accomplished via 10 Gbit/s ethernet connections. Communication with the scanner was performed using a 1 Gbit/s ethernet connection. ***Test:*** Our framework was tested on a 5-node configuration each with two GPUs: one of the nodes was reserved as the *master* node, and the remaining four nodes were assigned as reconstruction nodes. To evaluate our distributed MRI reconstruction framework, the following modules were implemented on the GPU as *services*: through-time radial GRAPPA (as described in (3)), convolution gridder, FFT operations, RSS combination/image cropping. Undersampled radial datasets acquired with 32 coils were reconstructed in real-time using our distributed MRI reconstruction framework. Four nodes were during (eight GPUs in total). ***MRI:*** MRI was performed on a 1.5T Espree scanner (Siemens Healthcare, Erlangen, Germany). Acquisition parameters were: radial acquisition matrix: 144x256 (calibration), 16x256 (accelerated), acceleration rate (R): 8, image matrix: 128x128, TR: 2.64ms, FOV: 300x300 mm^2, BW: 1115Hz/px, number of coils: 32, temporal resolution: 42 ms/frame. Imaging was performed with prior written informed consent and local IRB approval.

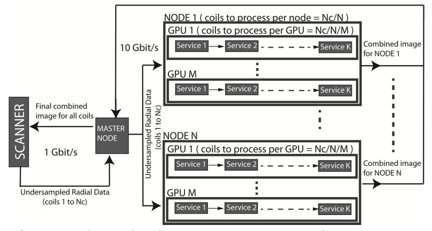
NODE 1 (coils to process per node = Nc/N)
GPU 1 (coils to process per GPU = Nc/N/M)
10 Gbit/s
Service 1
Service 2
Service K
Combined image for NODE 1
GPU M
SCANNER
Final combined image for all coils
MASTER NODE
1 Gbit/s
Undersampled Radial Data (coils 1 to Nc)
NODE N
Combined image for NODE N

Figure 1. Distributed reconstruction process. *Master* node forwards raw data to the reconstruction nodes. Each node then locally distributes its task to GPUs. Each GPU executes the same pipeline, but on different portions of the raw data. Partial results from each node combined by the *master* node and sent to scanner.

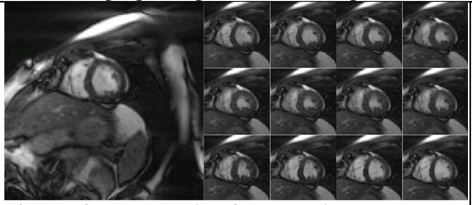

Figure 2. Short axis, free-breathing non-gated, 128x128 cardiac images from a healthy volunteer reconstructed from a 32-coil 16x256 radial data set using radial GRAPPA in real-time. Acquisition time: 42 ms, reconstruction time with 4 nodes (2 GPUs on each node): 11.2 ms.

	1 node	2 nodes	4 nodes
1 GPU	31.5	19	13.2
2 GPUs	19.5	13.4	11.2

Table 1. Through-time Radial GRAPPA reconstruction performances in ms, including network transfers between nodes (0.2-0.5 ms), from 32 coil, 16x256 radial data for various configurations.

RESULTS: Multi-node reconstruction performances on 32 coil 16x256 undersampled radial datasets are represented in Table 1 for various configurations. Short-axis, free-breathing, non-gated, systolic and diastolic cardiac images (R=8, 16 projections) are presented in Figure 2.

DISCUSSION: We present a generic, flexible, lightweight, distributed (multi-node, multi-GPU), low-latency reconstruction framework that is capable of providing faster-than-acquisition reconstruction performance. A 32 coil undersampled radial dataset (16x256 acquisition matrix, temporal resolution=42 ms) was reconstructed using radial GRAPPA in 11.2 ms when using 4 reconstruction nodes. Since the task distribution (both local and remote) is completely transparent to the user, our framework is scalable and can easily adapt to challenging reconstruction scenarios (e.g. higher acceleration rates, larger number of acquisition coils) by deploying more nodes/GPUs. Additionally, dynamic imaging methods could be readily supported, since asynchronous execution threads were transparently assigned to each *service*. As an added benefit, distributing the reconstruction task substantially reduced the memory requirements for each node/GPU. We believe that our framework is beneficial for reducing computational load and memory requirements for reconstruction on modern MRI scanners that support very large number of acquisition coils.

REFERENCES: **[1]** Santos JM et al. In Proc IEEE EMBS 2004.2:1048-51. **[2]** Hansen MS et al. MRM 2012 Epub. **[3]** Saybasili et al. Proc ISMRM 2012. p 2554.

FUNDING: This project was funded by NIH/NIBIB R00EB011527, and NIH 1RO1HL094557.