Published in final edited form as:
*Magn Reson Med*. 2015 March ; 73(3): 1015–1025. doi:10.1002/mrm.25213.

# Distributed MRI Reconstruction using Gadgetron based Cloud Computing

**Hui Xue**[1], **Souheil Inati**[2], **Thomas Sangild Sørensen**[3], **Peter Kellman**[4], and **Michael S. Hansen**[1]

[1]Magnetic Resonance Technology Program, National Heart, Lung and Blood Institute, National Institutes of Health, Bethesda, MD, USA

[2]National Institute of Mental Health, National Institutes of Health, Bethesda, MD, USA

[3]Department of Computer Science and Department of Clinical Medicine, Aarhus University, Aarhus, Denmark

[4]Medical Signal and Image Processing Program, National Heart, Lung and Blood Institute, National Institutes of Health, Bethesda, MD, USA

## Abstract

**Purpose:** To expand the Open Source Gadgetron reconstruction framework to support distributed computing and to demonstrate that a multi-node version of the Gadgetron can be used to provide non-linear reconstruction with clinically acceptable latency.

**Methods:** The Gadgetron framework was extended with new software components that enable an arbitrary number of Gadgetron instances to collaborate on a reconstruction task. This cloud-enabled version of the Gadgetron was deployed on three different distributed computing platforms ranging from a heterogeneous collection of commodity computers to the commercial Amazon Elastic Compute Cloud. The Gadgetron cloud was used to provide non-linear, compressed sensing, reconstruction on a clinical scanner with low reconstruction latency for example cardiac and neuro imaging applications.

**Results:** The proposed setup was able to handle acquisition and l1-SPIRiT reconstruction of nine high temporal resolution real-time, cardiac short axis cine acquisitions, covering the ventricles for functional evaluation, in under one minute. A three-dimensional high-resolution brain acquisition with 1 mm$^3$ isotropic pixel size was acquired and reconstructed with non-linear reconstruction in less than five minutes.

**Conclusion:** A distributed computing enabled Gadgetron provides a scalable way to improve reconstruction performance using commodity cluster computing. Non-linear, compressed sensing reconstruction can be deployed clinically with low image reconstruction latency.

### Keywords

Gadgetron; Distributed computing; Non-linear MRI reconstruction; Open-source software

**Corresponding author:** Hui Xue, Magnetic Resonance Technology Program, National Heart, Lung and Blood Institute, National Institutes of Health, 10 Center Drive, Bethesda, MD 20814, USA, Phone: +1 (301) 496-3052, Cell: +1 (609) 712-3398, Fax: +1 (301) 496-2389, hui.xue@nih.gov.

## Introduction

Image reconstruction algorithms are an essential part of modern medical imaging devices. The complexity of the reconstruction software is increasing due to the competing demands of improved image quality and shortened acquisition time. In the field of MR imaging, in particular, the image reconstruction has advanced well beyond simple fast Fourier transforms to include parallel imaging (1–5), non-linear reconstruction (6,7) and real-time reconstruction (8,9). The increased interest in 3D acquisitions and non-Cartesian sampling has increased the computational demands further. For applications where lengthy acquisition and reconstruction time is prohibited by physiological motion, biological processes, or limited patient cooperation, the reconstruction system is under further pressure to deliver images with low latency in order to keep up with the ongoing study.

While the need for fast image reconstruction is growing, most published reconstruction algorithms, especially those relying on iterative solvers, such as image domain compressive sensing (6,10–12) and *k*-space SPIRiT and its regularized versions (7,13,14), do not come with the efficient reference implementations that would enable clinical use. In many cases, the algorithms are not implemented for *online* use on clinical scanners. Even if the developers would like to integrate their reconstruction algorithms for *online* use, the vendor provided hardware and software platform may have inadequate specifications for a demanding reconstruction or the available programming window may be unsuited for integration of new reconstruction schemes. Consequently, there is a gap between the number of new algorithms being developed and published and the clinical testing and validation of these algorithms. Undoubtedly, this is having an impact on the clinical adoption of novel non-linear reconstruction approaches (e.g. compressed sensing).

We have previously introduced an open-source platform for medical imaging reconstruction algorithms called the Gadgetron (15), which aims to partially address the above-mentioned concerns. This platform is freely available to the research community and industry partners. It is platform independent and flexible for both prototyping and commercial development. Moreover, interfaces to several commercial MR platforms have been developed and are being shared in the research community. This simplifies the *online* integration of new reconstruction algorithms significantly and the new algorithms in research papers can be tested in clinical settings with less implementation effort. As a result, some groups have used Gadgetron for *online* implementation and evaluation of their reconstruction methods (16–18). Since the publication of the first version of the Gadgetron, the framework has adopted a vendor independent raw data format, the ISMRM Raw Data (ISMRMRD) format (19). This further enables sharing of reconstruction algorithms.

While this concept of an open-source platform and a unified ISMRMRD format shows great potential, the original Gadgetron framework did not support distributed computing across multiple computational nodes. Although the Gadgetron was designed for high performance (using multiple cores or GPU processors), it was originally implemented to operate within a single node or process. Distributed computing was not integral to the design. As reconstruction algorithms increase in complexity they may need computational power that would not be economical to assemble in a single node. The same considerations have led to the development of commodity computing clusters where a group of relatively modest computers are assembled to form a powerful computing cluster. An example of such a cluster system is the National Institutes of Health Biowulf Cluster (http://biowulf.nih.gov). Recently commercial cloud based services also offer the ability to configure such commodity computing clusters on demand and rent them by the hour. The Amazon Elastic Compute Cloud (EC2) is an example of such a service (http://aws.amazon.com/ec2).

In this paper, we propose to extend Gadgetron framework to enable cloud computing on multiple nodes. With this extension (named "Gadgetron Plus or GT-Plus"), any number of Gadgetron processes can be started at multiple computers (referred to as 'nodes') and a dedicated inter-process controlling scheme has been implemented to coordinate the Gadgetron processes to run on multiple nodes. A large MRI reconstruction task can be split and run in parallel across these nodes. This extension to distributed computing maintains the original advantages of Gadgetron framework. It is freely available and remains platform independent. As demonstrated in this paper, the nodes can even run different operating-systems (e.g. Windows or different distributions of Linux) and have different hardware configurations.

The implemented architecture allows the user to set up a GT-Plus cloud in a number of different ways. Specifically, it does not require a dedicated professional cloud computing platform. The GT-Plus cloud can be deployed on setups ranging from an arbitrary collection of networked computers in a laboratory (we refer to this as a "casual cloud") to high end commercial cloud systems, as demonstrated in the following. In this paper, we demonstrate the GT-Plus cloud set up in three different scenarios and demonstrate its flexibility to cope with variable computational environments. The first cloud setup is a "casual cloud", consisting of seven personal computers situated in our laboratory at the National Institutes of Health. The second setup uses NIH's Biowulf Cluster (20). The last configuration is deployed on the commercial Amazon Elastic Compute Cloud (Amazon EC2) (21).

To demonstrate the benefits of this extension, we used the cloud enabled version of the Gadgetron to implement nonlinear l1-SPIRiT (7) for 2D time resolved (2D+t) and 3D imaging applications. We demonstrate that cardiac cine imaging with 9-slices covering the entire left ventricle (32 channels, acquired temporal resolution 50ms, matrix size 192×100, acceleration factor 5, ~1.5s per slice) can be reconstructed with l1-SPIRiT with a reconstruction time (latency <30s) that is compatible with clinical workflow. Similarly we demonstrate high resolution 3D isotropic brain scans (20 channels, matrix size 256×256×192, acceleration factor 3×2, 1mm$^3$ isotropic acquired resolution), can be reconstructed with non-linear reconstruction with clinically acceptable latency (<2.5mins). For both cases, significant improvement of image quality is achieved with non-linear reconstruction compared to the linear GRAPPA results.

In the following sections, details of GT-Plus design and implementation are provided. The referral to specific source code components such as C++ classes, variables, and functions is indicated with monospaced font, e.g. GadgetCloudController.

## Methods

### Architecture and Implementation

In the following sections, we will first briefly review the Gadgetron architecture and the extensions that have been made to this architecture to enable cloud computing. Subsequently, we will describe two specific types of MRI reconstructions (2D time resolved imaging and 3D imaging) that have been deployed on this architecture.

**Gadgetron framework—**The Gadgetron framework is described in detail in (15). Here we briefly review the dataflow for comparison to the cloud based dataflow introduced below. As shown in Fig. 1, a Gadgetron reconstruction process consists of three components: Readers, Writers and Gadgets. A Reader receives and de-serializes the incoming data sent from the client (e.g. a client can be the MR scanner). A Writer serializes the reconstruction results and sends the data packages to the client. The Gadgets are connected to each other in a streaming framework as processing chains.

The Gadgetron maintains the communication with clients using a TCP/IP socket based connection. The typical communication protocol of Gadgetron process is the following:

**a.** The client issues the connection request to the Gadgetron server at a specific network port.

**b.** The Gadgetron accepts the connection and establish the TCP/IP communication.

**c.** The client sends an XML based *configuration* file to the Gadgetron server. This XML configuration outlines the Reader, Writers, and Gadgets to assemble a Gadget chain.

**d.** The Gadgetron server loads the required Readers, Writers and Gadgets from shared libraries as specified in the configuration file.

**e.** The client sends an XML based *parameter* file to the Gadgetron. The Gadgetron can initialize its reconstruction computation based on the parameters (e.g. acquisition matrix size, field-of-view, acceleration factor etc.). For MRI this XML parameter file is generally the ISMRMRD XML header describing the experiment.

**f.** The client sends every readout data to the Gadgetron in the ISMRMRD format. These data are de-serialized by the Reader and passed through the Gadget chain.

**g.** The reconstruction results are serialized by the Writer and sent back to the client via the socket connection.

**h.** When the end of acquisition is reached, the Gadgetron closes down the Gadget chain and closes the connection when the last reconstruction result has been passed back to the client.

**Gadgetron Plus (GT-Plus): distributed computing extension of Gadgetron—**A schematic outline of GT-Plus extension is shown in Fig. 2. A distributed Gadgetron process has at least one Gadgetron running on a specific port for each node (multiple Gadgetron processes can run within the same node at different ports). A software module, GadgetCloudController, manages the communication between nodes. Typically, the gateway node is receiving the readout data from the client and de-serializes them using a Reader. Depending on the reconstruction workflow, the gateway node may buffer the incoming readouts and perform some processing before sending reconstruction jobs to the connected nodes, or data can be forwarded directly to the client nodes. The GadgetCloudController maintains multiple TCP/IP socket connections with every connected node via a set of GadgetronCloudConnector objects (one for each connected node). Each GadgetronCloudConnector has a reader thread (CloudReaderTask) and a writer thread (CloudWriterTask) which are responsible for receiving and sending data (to the node) respectively. There is a Gadgetron Gadget chain running on every connected node. The gateway node will send the XML configuration to the connected node to assemble the chain. For different cloud nodes, different Gadget chains can be assembled. In fact, the connected nodes can also be gateway nodes, thus creating a multi-tiered cloud. The typical protocol of GT-Plus distributed computing is as follows:

**a.** The client issues the connection request to the GT-Plus gateway at a specific network port. Once the connection is established, the client will send the XML based configuration and parameter files to establish and initialize the gadget chain at the gateway node.

**b.** The client starts sending readout data to the gateway node.

**c.** The Gateway node establishes connection to cloud nodes and supplies them with XML configuration and parameter files. As mentioned, different connected nodes can be configured with different chains if needed.

**d.** Job packages are sent to connected cloud nodes for processing via the corresponding CloudWriterTask.

**e.** When all jobs are sent to nodes (the acquisition is done), the gateway Gadgetron either waits for reconstruction results or conducts extra computation. The ReaderTask objects, at the same time, listen for reconstruction results. Whenever the reconstruction results are sent back from a cloud node, the gateway Gadgetron is notified by the ReaderTask object and will take user-defined actions, e.g. passing the results downstream or waiting for the completion of all jobs. Finally the gateway node will proceed to finish other processing steps if any and send the images down the remaining part of the gateway Gadget chain and eventually back to the client.

**f.** If one or more connected nodes fail to complete a job successfully, the gateway node will be notified by either receiving an invalid result package or detecting a shutdown message on the socket. The GadgetCloudController on the gateway node will keep a record of uncompleted jobs and process them locally. In this way, the GT-Plus gains robustness against network instability.

The software modules mentioned above are implemented as C++ template classes and are capable of handling any type of custom job packages, i.e. the user is able to configure what data types are sent and received from the cloud nodes. The only requirement is to implement appropriate functions for serialization and de-serialization of work packages and results. This design is a straightforward extension of the Readers and Writers in the original Gadgetron.

In the current implementation, every computing node can be given an index indicating its computational power. This index is used to allow the gateway node to apply a scheduling algorithm where the workload is distributed to the cloud nodes in proportion to their computational capabilities. This can be important if the cloud consists of a heterogeneous set of hardware configuration.

To supply the network connection information of cloud nodes to the Gadgetron, the user can specify IP addresses or hostnames of nodes in the gateway XML configuration file. Alternatively, the framework can read in a cloud structure definition file.

The parallel computing and communication protocols used in this implementation are based directly on the existing communications protocols within the Gadgetron. In the previous versions of the software, the Gadgetron process itself was seen as a server, but in the current version, Gadgetron processes can act as clients too and connect to addition Gadgetron processes. Like the previous version of Gadgetron software, the Adpative Communication Environment (ACE, (22)) package is used for TCP/IP network communication between clients and servers. The connection between gateway and computing nodes are maintained using an ACE socket on certain port. Whenever a job package is to be sent to a node, the corresponding CloudWriterTask takes the package and forwards it on the socket to a connected Gadgetron node. The ReaderTask on the gateway node is always listening to the tcp/ip port for reconstruction results. Whenever a result package arrives, the ReaderTask receives it notifies the GadgetCloudController to process the incoming result package. To ease the thread synchronization between ReaderTask/WriterTask and GadgetCloudController, all three components are implemented as the active objects (23) with a message queue attached to each as the buffer for inbound/outbound communication.

**Gadgetron Plus (GT-Plus) for 2D time resolved (2D+t) Reconstruction Tasks—** In addition to the software modules for cloud communication, the GT-Plus extensions also include a specific job type to support 2D+t reconstruction tasks, e.g. multi-slice cine imaging. This workflow is used here to illustrate a cloud setup with a dual layer topology, as illustrated in Fig. 3. The gateway gadget chain will buffer readouts for a specified ISMRMRD dimension (e.g. for the multi-slice cine, it is usually the slice dimension). Once all data for a slice have arrived, the GtPlusRecon2DTGadgetCloud gadget will forward the job package to a first layer node to start the reconstruction.

For a 2D+t dynamic imaging task, one slice will have multiple 2D *k*-spaces. For example, for the cardiac cine acquisition, multiple cardiac phases are usually acquired. The first layer node responsible for the reconstruction of a given slice can choose to further distribute the dataset to a set of sub-nodes in the second layer. The first-layer nodes can serve solely to distribute jobs or they can perform computation as well. In principle, a given reconstruction job can utilize an arbitrary number of node layers to form a more complicated cloud topology.

**Gadgetron Plus (GT-Plus) for 3D Reconstruction Tasks—**For 3D acquisitions, a single layer cloud topology was used in this paper. The gateway node's GtPlusRecon3DTGadget receives the 3D acquisition and performs the processing such as coil compression and estimation of *k*-space convolution kernel for parallel imaging. It then splits the large 3D reconstruction problem by performing a 1D inverse FFT transform along the readout direction. Thus, the reconstruction is decoupled along the readout direction. Every chunk of data along the readout direction is then sent to a connect node for non-linear reconstruction. The gateway node will wait for all jobs to complete and results to return. It then reassembles the 3D volume from all chunks and continues to perform other post-processing, e.g. *k*-space filtering tasks and finally returns images to the client.

## Toolbox Features

The Gadgetron is divided into Gadgets and *Toolbox* algorithms that can be called from the Gadgets or standalone applications. In addition to the toolbox features listed in (15), the GT-Plus extensions add additional toolboxes. Here is an incomplete list of key algorithm modules:

**2D/3D GRAPPA.—**A GRAPPA implementation for 2D and 3D acquisition is added. It fully supports the ISMRMRD data format and different parallel imaging modes (embedded, separate or interleaved auto-calibration lines, as defined in (19)). For the 3D GRAPPA case, the implementation supports two-dimensional acceleration. To support parallel imaging in interactive or real-time applications, a real-time high-throughput 2D GRAPPA implementation using GPU is also provided in the Gadgetron.

**2D/3D Linear SPIRiT.—**A linear SPIRiT (7) reconstruction is implemented in the Gadgetron toolbox. Specifically, if the *k*-space $\boldsymbol{x}$ consists of filled points $\boldsymbol{a}$ and missing points $\boldsymbol{m}$ then:

$$\boldsymbol{x} = \boldsymbol{D}^{T}\boldsymbol{a} + \boldsymbol{D}_{c}^{T}\boldsymbol{m} \quad (1)$$

Here $\boldsymbol{x}$ is an vector containing *k*-space points for all phase encoding lines for all channels; $\boldsymbol{a}$ stores the acquired points, and $\boldsymbol{m}$ is for missing points. $\boldsymbol{D}$ and $\boldsymbol{D_c}$ are the sampling pattern matrixes for acquired and missing points, Linear SPIRiT solves the following equation:

$$(\boldsymbol{G} - \boldsymbol{I})\boldsymbol{D}_{c}^{T}\boldsymbol{m} = -(\boldsymbol{G} - \boldsymbol{I})\boldsymbol{D}^{T}\boldsymbol{a} \quad (2)$$

Here $\boldsymbol{G}$ is the SPIRiT kernel matrix, which is computed from a fully sampled set of auto calibration data.

**2D/3D Non-linear l1-SPIRiT.—**Equation 2 is extended by the L1 term:

$$argmin_{\boldsymbol{m}}\left\{\left\|(\boldsymbol{G} - \boldsymbol{I})\left(\boldsymbol{D}^{T}\boldsymbol{a} + \boldsymbol{D}_{c}^{T}\boldsymbol{m}\right)\right\|_{2} + \lambda\left\|\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{C}^{H}\boldsymbol{F}^{H}\left(\boldsymbol{D}^{T}\boldsymbol{a} + \boldsymbol{D}_{c}^{T}\boldsymbol{m}\right)\right\|_{1}\right\} \quad (3)$$

Another variation of l1SPIRiT is to treat the full *k*-space as the unknowns:

$$argmin_{\boldsymbol{x}}\left\{\left\|(\boldsymbol{G}-\boldsymbol{I})\boldsymbol{x}\right\|_2+\lambda\left\|\boldsymbol{W\Psi C^H F^H x}\right\|_1+\beta\left\|\boldsymbol{Dx}-\boldsymbol{a}\right\|_2\right\}. \quad (4)$$

Here $\boldsymbol{\Psi}$ is the sparse transform and $\boldsymbol{W}$ is an extra weighting matrix applied on the computed sparse coefficients to compensate for non-isotropic resolution or temporal redundancy. $\boldsymbol{F}$ is the Fourier transform matrix. $\boldsymbol{C}$ is the coil sensitivity.

**Redundant 2D and 3D wavelet transform.—**The redundant wavelet transform for 2D and 3D data arrays are implemented in the toolbox. It is used in the L1 regularization term. A fast implementation for redundant Harr wavelet is also provided.

## Example Applications

Corresponding to the two types of cloud topologies described in the previous sections, two *in vivo* experiments were performed on healthy volunteers. The local Institutional Review Board approved the study, and all volunteers gave written informed consent.

**Real-time multi-slice myocardial cine imaging using l1-SPIRiT—**The aim was to make it clinically feasible to assess myocardial function using real-time acquisitions and non-linear reconstructions covering the entire ventricles. With conventional reconstruction hardware, the reconstruction time for such an application would prohibit clinical adoption. The dual layer cloud topology was used here and every slice was sent to a separate node (first layer) which further split cardiac phases into multiple chunks. While processing one chunk itself, the first layer node also sent others to its sub-nodes for parallel processing. The algorithm workflow was as follows: a) Undersampled *k*-space data were acquired with the time-interleaved sampling pattern. b) The gateway node received the readout data and performed the on-the-fly noise pre-whitening (5). c) The data from one slice was sent to one first layer node. d) To reconstruct the underlying real-time cine images, the auto-calibration signal (ACS) data for a slice were obtained by averaging all undersampled *k*-space frames at this slice. e) The SPIRiT kernel was estimated on the assembled ACS data. f) The data for the slice was split into multiple chunks and sent to sub-nodes, together with the estimated kernel. The size of a data chunk for a node was proportional to its computing power index. f) Sub-nodes received the data package and solved equation 3. The linear SPIRiT problem (equation 2) was first solved to initialize the non-linear solver. g) Once the process for a slice was completed, the node sent the reconstructed frames back to gateway, which then returned them to the scanner. Note the reconstruction for a given slice started while acquisition was still ongoing for subsequent slices. Thus the data acquisition and processing was overlapped in time to minimize the overall waiting time after the data acquisition.

Fig. 4 outlines the l1SPIRiT reconstruction gadget chain for multi-slice cine imaging using the dual layer cloud topology. The raw data first went through the NoiseAdjustGadget, which performed noise pre-whitening. After removing the oversampling along the readout direction, the data was buffered in the AccumulatorGadget. Whenever the acquisition for a slice was complete, the data package was sent to the GtPlusRecon2DTGadgetCloud gadget, which established the network connection to the first layer nodes and sent the data. The first layer node ran only the GtPlusReconJob2DTGadget gadget, which then connected to the second layer nodes. The CloudJobReader and CloudJobWriter were used to serialize and de-serialize the cloud job package.

Imaging experiments were performed on a 1.5T clinical MRI system (MAGNETOM Aera, Siemens AG Healthcare Sector, Erlangen, Germany) equipped with a 32-channel surface coil. A healthy volunteer (female, 23.8yrs) was scanned. Acquisition parameters for free-breathing cine were as follows: balanced SSFP readout, TR = 2.53/TE = 1.04ms, acquired matrix size 192×100, flip angle 60°, FOV 320×240mm$^2$, slice thickness 8mm with a gap of 2mm, bandwidth 723 Hz/pixel, interleaved acquisition pattern with acceleration factor R=5. The whole left ventricular was covered by 9 slices and acquisition duration for every slice was ~1.5s with one dummy heartbeat between slices. The scan time (defined as the time to perform data acquisition) to complete all 9 slices was 22.6s.

**High resolution neuro imaging using l1-SPIRiT—**The second example aimed to use GT-Plus for non-linear reconstruction of a high resolution 3D acquisition. The algorithm workflow was as follows: a) Undersampled *k*-space data were acquired with the fully sampled center region. b) The gateway node received the readout data and performed the on-the-fly noise pre-whitening. c) When all data was received, the SPIRiT kernel calibration was performed on the fully sampled ACS region. The kernel was zero-padded only along the readout direction and Fourier transformed to the image domain. This was done to reduce the maximal memory needed to store the image domain kernel and decrease the amount of data transferred over the network to connected nodes. The *k*-space data was transformed to image domain along the readout as well. d) The gateway node split the image domain kernel and data into multiple chunks along the readout direction and sent them to multiple connected nodes. e) The connected nodes received the packages and zero-padded kernels along the remaining two spatial dimensions and transformed into image domain. The image domain kernel was applied to the aliased images by pixel-wise multiplication. This linear reconstruction was performed to initialize the non-linear reconstruction. f) After receiving all reconstructed images from connected nodes, the gateway assembled the 3D volume and performed post-processing, such as *k*-space filtering and then sent results back to the scanner.

The imaging experiments were performed on a 3.0T clinical MRI system (MAGNETOM Skyra, Siemens AG Healthcare Sector, Erlangen, Germany) equipped with a 20-channel head coil. The acquisition parameters were as follows: GRE readout, TR = 10.0/TE = 3.11ms, acquired matrix size 256×256×192, flip angle 20°, isotropic spatial resolution 1mm$^3$, bandwidth 130 Hz/pixel, two dimension acceleration factor R=3×2. The embedded parallel imaging mode was used with the ACS signal acquired as a 32×32 fully sampled region. The total acquisition time was 91.6s.

## Deployment and Scanner Integration

The cloud extension of Gadgetron (GT-Plus), can be deployed on different types of platforms. Three setups have been tested here for *in vivo* experiments and the GT-Plus software can be deployed on those setups without any code changes. The first setup (referred to as the "casual cloud") is a heterogeneous collection of computers, e.g. as one would find in many MRI research laboratories. These computers have different hardware and operating system configurations. The second setup is the NIH Biowulf cluster, which is a custom built cluster with totally 2300 nodes (>12000 computing cores); it is a shared system and users request a specific amount of resources for a given computational task. The third setup is the Amazon Elastic Compute Cloud (EC2), where an arbitrary amount of computational power can be rented by the hour providing the flexibility to tailor the cloud configuration to suit a specific application.

**"Casual" cloud—**The first setup is a solution that almost any MRI research laboratory would be able to use by installing the Gadgetron on a set of networked computers and using one of these computers as the gateway node. No specific operating system or special hardware is needed. Here, six personal computers on the NIH intranet (1Gb/s connections) were used as the cloud nodes and two more computers were used as gateway nodes for 2D+t and 3D experiments respectively. The Gadgetron software was compiled and installed on all computers. The gateway node for 2D+t test was a desktop computer, running windows 7 Pro 64 bit (four core Intel Xeon E5–2670 2.60GHz processers, 48GB DDR3 RAM). The gateway node for 3D test also ran the same operating-system (two eight-core Intel Xeon E5–2670 2.60GHz processers, 192GB DDR3 RAM).

Among the six cloud nodes, two of them were running windows 7 Pro 64bit (each had four cores of Intel Xeon E5–2670 2.60GHz, and 48GB DDR3 RAM). Ubuntu linux 12.04 were running on other four nodes (two nodes each had six cores of Intel Xeon E5645 2.4GHz and 24GB DDR3 RAM; the other two had four cores of Intel Xeon X5550 2.67GHz and 24GB DDR3 RAM).

For the dual layer cloud test, one Windows and two Ubuntu computers served as first layer nodes. The other three nodes were on the second layer. Each of them was connected to one first layer node. For the 3D reconstruction test, all six nodes were connected directly to the gateway.

**NIH Biowulf cloud—**The second cloud setup tested in this study utilized the NIH Biowulf cluster. NIH Biowulf system is a GNU/Linux parallel processing system designed and built at the National Institutes of Health. Biowulf consists of a main login node and a large number of computing nodes. The computing nodes within Biowulf are connected by a 1Gb/s network. The MRI scanner used in this study was also connected to Biowulf system with a 1Gb/s network connection. For the 2D multi-slice cine experiments 37 nodes were requested from the cluster. The gateway node had 16 cores (two eight-core Intel Xeon E5–2670 2.60GHz processers) and 72GB DDR3 RAM. All other nodes had identical configurations (two six-core Intel Xeon X5660 2.80GHz, 24GB DDR3 RAM). For the dual layer cloud test, 9 nodes were used on the first layer to match the number of acquired slices. Each of them was connected to three sub-nodes on the second layer. For the 3D reconstruction test, the same gateway node and 23 connected nodes were requested.

The cloud topology was selected to balance the maximal parallelization and programming complexity. It is convenient to have dual layer structure for the multi-slice cine imaging, because this structure allows overlap between data acquisition and reconstruction computation. For the 3D acquisition, the reconstruction only starts when the data acquisition is completed; therefore, a simple one layer cloud is used to simplify the synchronization. The number of nodes used in the study was mainly limited by the available resources during the experiments. Although Biowulf system has a large number of nodes, hundreds of jobs from multiple users can run in parallel at any given time. We also intentionally made number of nodes different between 2D+t and 3D tests, to challenge the scalability of Gadgetron based cloud software.

**Amazon EC2 based cloud—**The last setup utilized the Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 cloud provides resizable computing capacity in the Amazon Web Services (AWS) cloud. User can configure and launch flexible number of computing nodes. Every node can have different hardware configuration and operating system. Amazon EC2 provides a broad selection of Windows and Linux operating systems. More details about Amazon EC2 can be found in (21).

For this study, 19 nodes were launched to build up a virtual cloud on Amazon EC2. Every node used the cc2.8xlarge instance type (two eight-core Intel Xeon E5–2670 2.60GHz processers, 20MB L3 cache per processor, 60GB DDR3 RAM), running Ubuntu Server 13.10. All nodes were connected 10Gb/s connections. For the dual layer test, one node was used as the gateway and 9 nodes were on the first layer. Each first layer nodes were connected to two other sub-nodes. Thus, data for every slice was processed by three nodes in total. For the 3D reconstruction test, all 18 nodes are connected directly to the gateway. At the time of the study, the price for the nodes used in this test was US $2.4 per hour per node or US $45.60 per hour for the complete cloud setup. In this case, the number of nodes was chosen as a reasonable balance between cost and performance.

The connection speed from the MRI scanner to the Amazon EC2 cloud was measured to be 49.8MB/s or 0.39Gb/s at the time of the experiments.

**Clinical scanner integration—**As previously described, the Gadgetron enables direct integration with the MRI scanner for online reconstruction (15). The cloud extension of GT-Plus maintains this advantage of *online* processing. Effectively, this setup enables a seamless connection of cloud computing resources directly to the scanner. From the end-user perspective, the only noticeable difference between reconstruction in the cloud and locally (on the scanner's reconstruction hardware) is the improvement in reconstruction speed.

The integration of cloud computing resources with a clinical scanner raises questions about patient privacy. In this study multiple steps were taken to mitigate any risks. Firstly, all data being sent to the Gadgetron from the scanner was anonymized. Absolutely no identifying information that could be used to trace back to the patient were sent over the network. The only information sent was the MR raw data and protocol parameters such as field of view, matrix size, bandwidth, etc. Secondly, all data transfer between scanner and gateway node was encrypted via a secure shell (SSH) tunnel (24). All other nodes connected to the gateway were behind firewalls. Besides serving as a security measure, the use of SSH tunnels to connect to the Gadgetron was a convenient way to test multiple Gadgetron setups on the same scanner. The reconstruction could simply be directed to a different Gadgetron by changing the SSH tunnel pointing to a different Gadgetron instance.

## Results

### Multi-slice myocardial cine imaging

Fig. 5 shows the reconstruction results generated by the GT-Plus cloud, illustrating the noticeable improved in image quality using non-linear reconstruction.

Table 1 shows the total imaging time and computing time. The imaging time is the time period from the start of data acquisition to the moment when images of all slices were returned to scanner. The computing time is defined as the time used to perform the reconstruction computation. On the described Amazon EC2 cloud, the total imaging time was 52.6s. The computing time was 48.9s because the computation was overlapped with the data acquisition. On the NIH's Biowulf cloud, imaging and computing times were 62.1s and 58.2s respectively. If only a single node was used, the computing time was 427.9s and 558.1s for two cloud setups. Therefore, the multi-slice cine imaging with entire ventricle coverage was completed within 1min using the non-linear reconstruction on the GT-Plus cloud. The casual cloud gave computing time of 255.0s and if only one node was used, the time went up to 823.7s. This speedup may be helpful to the development and validation of non-linear reconstruction in a MRI research lab.

### High resolution 3D neuroimaging

Fig. 6 illustrates the cloud reconstruction results using l1-SPIRiT. Compared to the GRAPPA, the non-linear reconstruction shows noticeable SNR improvements. Table 2 gives the total imaging time and computing time for three cloud setups and the single node processing. The total imaging time on the Biowulf cloud was 278.5s which includes the computing time of 186.4s, since not all computational steps ran in parallel for this 3D reconstruction. As every computing node of the Amazon EC2 cloud setup had 16 cores and a newer CPU model, the total computing time was further reduced to 146.0s. Total imaging time on the Amazon cloud was 255.0s. Thus the latency following data acquisition was less than 2mins.

### Availability and Platform support

The entire package is integrated with the currently available version of the Gadgetron, which is distributed under a permissive, free software license based on the Berkeley Software Distribution license. The licensing terms allow users to use, distribute and modify the software in source or binary form. The source code of entire Gadgetron package (including the gtPlus extensions) can be downloaded from http://gadgetron.sourceforge.net/. Thecodebase used in this article corresponds to release 2.5.0, or the git code repository commit with SHA-1 hash 415d999aa83860225fdddc0e94879fc9c927d64d. The documentation can be found at the Sourceforge Open Source distribution website for Gadgetron (http://gadgetron.sourceforge.net/). The software has been compiled and tested on

Microsoft Windows 7 64bit, Ubuntu Linux, CentOS 6.4, and Mac OS X (Note this paper does not present examples running on CentOS and Mac OS X, but the software has been compiled and tested on those two operating systems).

## Discussion

This work extends the previously published Gadgetron framework to support distributed computing, which makes it possible to distribute a demanding reconstruction task over multiple computing nodes and significantly reduce the processing time. This paper focused on reducing the processing time of non-linear reconstruction applications, which have so far been difficult to deploy clinically. As demonstrated in the examples, the increased computational power could make the processing time of non-linear reconstruction feasible for the regular clinical use.

The Gadgetron Plus (GT-Plus) extension provides a set of utility functions to manage and communicate with multiple nodes. Based on these functions, flexible cloud topologies can be realized as demonstrated in this paper. In the current Gadgetron software release, the discussed dual-layer and single layer cloud implementation are made available to the community. They are designed and implemented for general-purpose use, although the examples given are specific for cine and 3D neuroimaging. Based on the GT-Plus extensions new cloud topologies will be explored in the future and users can design and implement their own cloud structure.

The Gadgetron based cloud computing was deployed on three different types of cloud computing hardware. This demonstrates the flexibility of both Gadgetron framework in general and the GT-Plus extensions specifically. The casual cloud setup is the cheapest, most accessible way of using Gadgetron cloud computing. It can be set up quickly and used for algorithm development and validation in a research laboratory. The disadvantage of using such a setup is the computing nodes may be very heterogeneous and optimal distribution of the work among nodes may become non-trivial. If cloud computing is used in a production type setting on a clinical scanner, it is also problematic that a node may be busy with other tasks when it is needed by the scanner.

The Amazon EC2 system, on the other hand, provides a professional grade cloud setup with 24/7 availability. The computational resources can be dedicated for a specific project or scanner and the nodes can in general have the same configuration, which reduces the complexity of scheduling. This setup is also easy to replicate and share among groups and scanners. There are other companies that provide the same type of cloud infrastructure service, e.g. Rackspace (http://www.rackspace.com/). With these vendors the users pay by hour to rent computers. In the setup we describe in this paper, the cost of running the cloud was on the order of US $50 per hour. While this cost could be prohibitive if the cloud is enabled 24 hours per day, it is a relatively modest cost in comparison to the cost typical patient scans. Moreover, this cost is falling rapidly with more service providers entering into the market and more powerful computing hardware becoming available. The main downside of using a remote cloud service such as Amazon EC2 is the need for a high-speed Internet connection to the cloud provider. At large universities 1Gb/s connections (which are sufficient for the applications presented here) are becoming commonplace. However, this may not be the case for hospitals in general.

The third setup, the NIH's Biowulf, is a dedicated, custom built, cluster system. While not available to the entire MRI community, it represents a parallel computing platform often found at research institutions (such as universities) or large corporations. For large organizations aiming to provide imaging, reconstruction and processing service for a large number of scanners, this setup may be the most cost effective platform. It is, however, important to note that purchasing, configuration, deployment, and management of even modest cloud computing resources is a non-trivial task that requires significant resources.

Several software modules implemented in the Gadgetron framework are GPU accelerated, although the demonstrated examples in this paper are mainly using CPUs. The GT-Plus extension of distributed computing does not conflict with GPU based computational acceleration in any sense. Every computing node can be equipped with different hardware; for those with good GPU processers or CPU coprocessors (e.g. Intel Xeon Phi coprocessors http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html), the local processing can utilize those resources, since the Gadgetron framework allows every node to be configured with different Gadget chains. Heterogeneous computational hardware across nodes can be challenging for optimal load balancing. In the current framework, the user can supply computing power indexes for nodes to indicate its strength. But the framework does not presently implement more complicated approaches to help determine the computing ability of every node.

The Gadgetron framework and its toolboxes are mainly programmed using C++ and fully utilize generic template programming. While efforts are made to provide documentation and examples in the Gadgetron distribution, developers who want to extend the toolboxes still need some proficiency with object oriented programming and C++ in particular.

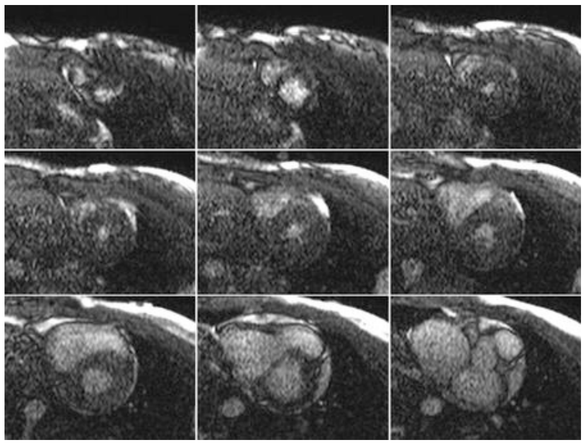
## Conclusion

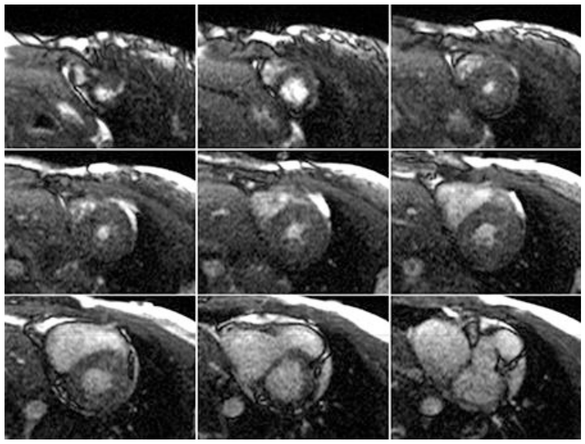
We have developed the Gadgetron Plus extension for the Gadgetron framework to support distributed computing using various cloud setups. The experiments using this distributed computing setup show that the increased computational power in the cloud can be used to accelerate l1-SPIRIT reconstructions significantly and thus make them clinically practical. These results indicate that the Gadgetron based can help with the clinical testing and adoption of advanced non-linear reconstruction methods. Furthermore, the open source nature of implemented software framework can help the advancement of reproducible research encouraging software sharing.

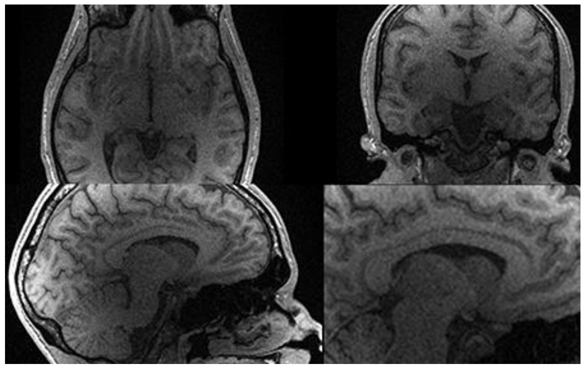
## ACKNOWLEDGEMENTS

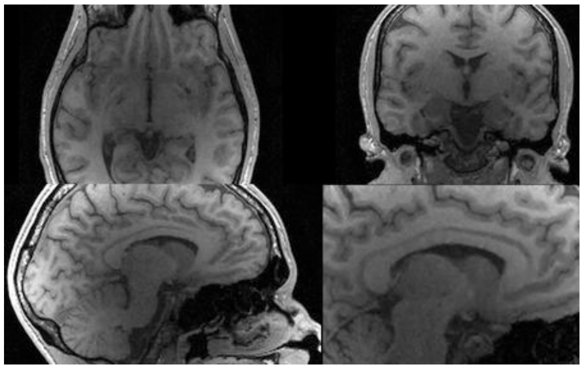
This research was supported fully by the Intramural Research Program of the NIH, National Heart, Lung and Blood Institute. This study utilized the high-performance computational capabilities of the Biowulf Linux cluster at the National Institutes of Health, Bethesda, Maryland, USA. (http://biowulf.nih.gov).

## Appendix









## References

1. Blaimer M, Breuer F, Mueller M, Heidemann RM, Griswold MA, Jakob PM. SMASH, SENSE, PILS, GRAPPA: How to Choose the Optimal Method. Topics in Magnetic Resonance Imaging 2004;15(4):223–236. [PubMed: 15548953]
2. Breuer FA, Kellman P, Griswold MA, Jakob PM. Dynamic autocalibrated parallel imaging using temporal GRAPPA (TGRAPPA). Magnetic Resonance in Medicine 2005;53(4):981–985. [PubMed: 15799044]
3. Griswold MA, Jakob PM, Heidemann RM, Nittka M, Jellus V, Wang J, Kiefer B, Haase A. Generalized autocalibrating partially parallel acquisitions (GRAPPA). Magnetic Resonance in Medicine 2002;47(6):1202–1210. [PubMed: 12111967]
4. Kellman P, Epstein FH, McVeigh ER. Adaptive sensitivity encoding incorporating temporal filtering (TSENSE). Magnetic Resonance in Medicine 2001;45(5):846–852. [PubMed: 11323811]
5. Pruessmann KP, Weiger M, Scheidegger MB, Boesiger P. SENSE: Sensitivity encoding for fast MRI. Magnetic Resonance in Medicine 1999;42(5):952–962. [PubMed: 10542355]
6. Lustig M, Donoho D, Pauly JM. Sparse MRI: The application of compressed sensing for rapid MR imaging. Magnetic Resonance in Medicine 2007;58(6):1182–1195. [PubMed: 17969013]
7. Lustig M, Pauly JM. SPIRiT: Iterative self-consistent parallel imaging reconstruction from arbitrary k-space. Magnetic Resonance in Medicine 2010;64(2):457–471. [PubMed: 20665790]
8. Hansen MS, Atkinson D, Sorensen TS. Cartesian SENSE and k-t SENSE reconstruction using commodity graphics hardware. Magnetic Resonance in Medicine 2008;59(3):463–468. [PubMed: 18306398]
9. Sorensen TS, Schaeffter T, Noe KO, Hansen MS. Accelerating the Nonequispaced Fast Fourier Transform on Commodity Graphics Hardware. Medical Imaging, IEEE Transactions on 2008;27(4): 538–547.
10. Jung H, Sung K, Nayak KS, Kim EY, Ye JC. k-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI. Magnetic Resonance in Medicine 2009;61(1):103–116. [PubMed: 19097216]
11. Liang D, DiBella EVR, Chen R-R, Ying L. k-t ISD: Dynamic cardiac MR imaging using compressed sensing with iterative support detection. Magnetic Resonance in Medicine 2011;68(1): 41–53. [PubMed: 22113706]
12. Lustig M, Santos JM, Donoho DL, Pauly JM. k-t SPARSE: High Frame Rate Dynamic MRI Exploiting Spatio-Temporal Sparsity. 2006 6–12 May; Seattle, Washington, USA p 2420.
13. Vasanawala SS, Alley MT, Hargreaves BA, Barth RA, Pauly JM, Lustig M. Improved Pediatric MR Imaging with Compressed Sensing. Radiology 2010;256(2):607–616. [PubMed: 20529991]
14. Vasanawala SS, Murphy MJ, Alley MT, Lai P, Keutzer K, Pauly JM, Lustig M. Practical parallel imaging compressed sensing MRI: Summary of two years of experience in accelerating body MRI of pediatric patients. 2011 March 30 - April 2; Chicago, IL p1039–1043.
15. Hansen MS, Sørensen TS. Gadgetron: An open source framework for medical image reconstruction. Magnetic Resonance in Medicine 2012;69(6):1768–1776. [PubMed: 22791598]

16. Feng Y, Song Y, Wang C, Xin X, Feng Q, Chen W. Fast direct fourier reconstruction of radial and PROPELLER MRI data using the chirp transform algorithm on graphics hardware. Magnetic Resonance in Medicine 2013;70(4):1087–1094. [PubMed: 23165973]

17. Simpson R, Keegan J, Gatehouse P, Hansen M, Firmin D. Spiral tissue phase velocity mapping in a breath-hold with non-cartesian SENSE. Magnetic Resonance in Medicine 2013:doi: 10.1002/mrm.24971.

18. Xue H, Kellman P, LaRocca G, Arai A, Hansen M. High spatial and temporal resolution retrospective cine cardiovascular magnetic resonance from shortened free breathing real-time acquisitions. Journal of Cardiovascular Magnetic Resonance 2013;15:102. [PubMed: 24228930]

19. Hansen MS. Sharing Reconstruction Software – Raw Data Format and Gadgetron. 2013; Sedona, AZ, USA.

20. NIH Biowulf cluster system. http://biowulf.nih.gov/. Bethesda, Maryland, USA July 15, 2013.

21. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/. July 12, 2013.

22. Schmidt DC. The ADAPTIVE communication environment: object-Oriented Network Programming Components for Developing Client/Server Applications. The 11th Annual Sun Users Group Conference San Jose, California, USA 1993 p 214–225.

23. Lavender RG, Schmidt DC. Active object: an object behavioral pattern for concurrent programming In: John MV, James OC, Norman LK, editors. Pattern languages of program design 2: Addison-Wesley Longman Publishing Co., Inc.; 1996 p 483–499.

24. Network Working Group of the IETF. The Secure Shell (SSH) Authentication Protocol. 2006.

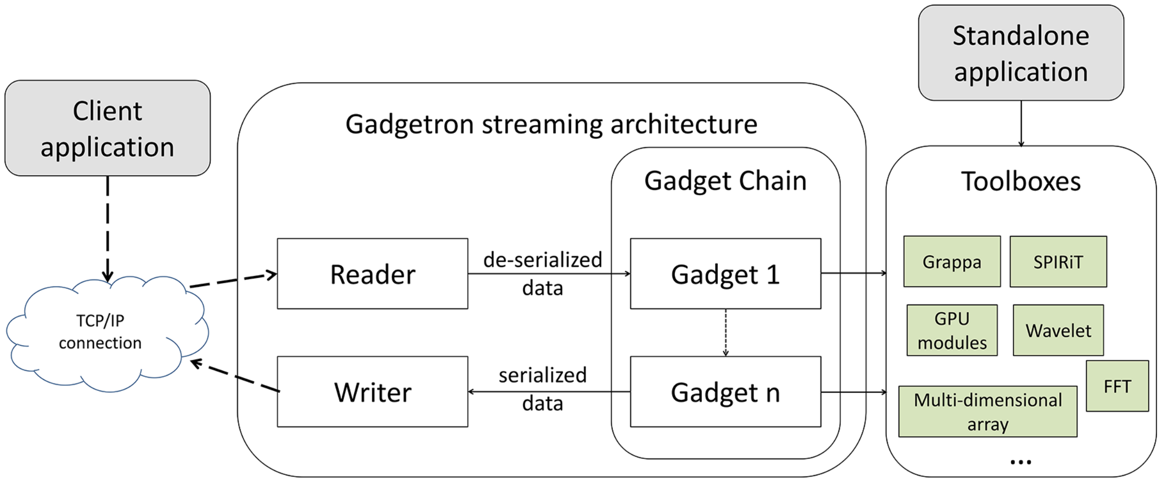


**FIG 1.**
A schematic presentation of Gadgetron architecture. The client application communicates with Gadgetron process via the TCP/IP connection. The Reader de-serializes the incoming data and passes them through the Gadget chain. The results are serialized by the Writer and sent back to client application. The Gadgetron process resides on one computer, the Gadgetron server. The client can be a MRI scanner or other customer processes. Note the algorithm modules implemented in the Gadgetron toolboxes are independent from the streaming framework; therefore, these functionalities can be called by standalone applications which are not using the streaming framework.

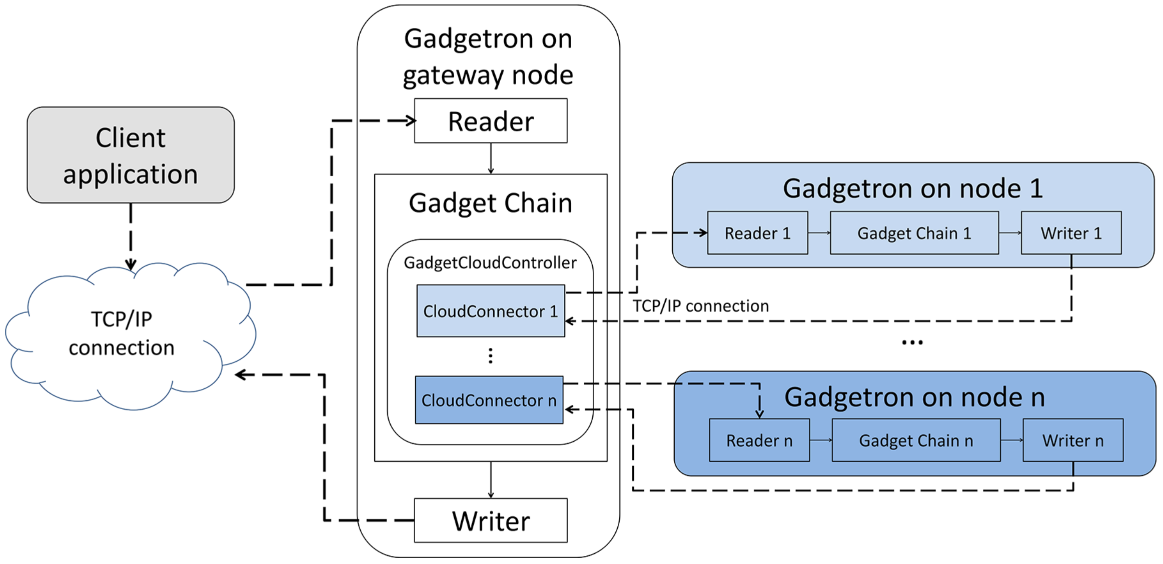


**FIG 2.**
Example presentation of GT-Plus distributed computing architecture for Gadgetron. In this example, at least one Gadgetron process is running on a node (Multiple Gadgetron processes can run in one node on different ports). The gateway node communicates with the client application (e.g. MRI scanner). It manages the connections to each of the computing nodes via the software module GadgetCloudController and GadgetronCloudConnector. Whenever sufficient data is buffered on the gateway node, the package can be sent to the computing node for processing. Different computing nodes can run completely different reconstruction chain.

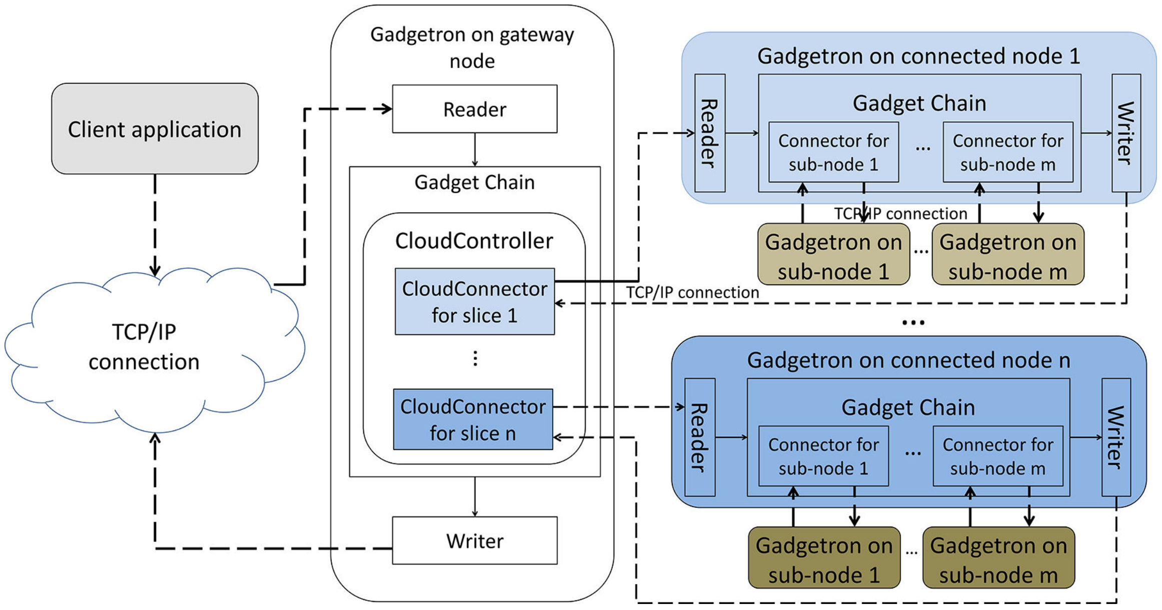


**FIG 3.**
The dual layer topology of cloud computing for 2D+t applications. Every connected node contains its own GadgetCloudController and can split a reconstruction task for a slice to its sub-nodes. Compared to the basic cloud topology shown in Fig. 2, this dual layer example adds extra complexity. This example demonstrates the flexibility of GT-Plus architecture for composing different computing cloud topologies to fit different reconstruction tasks.

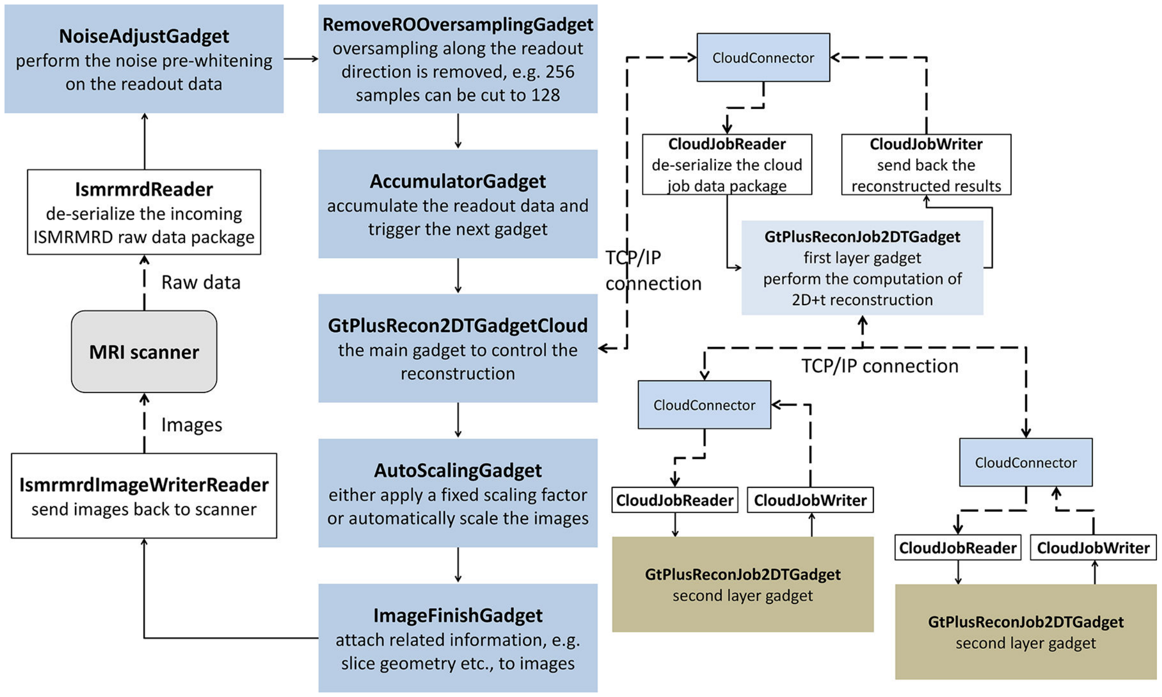

NoiseAdjustGadget
perform the noise pre-whitening on the readout data
RemoveROOversamplingGadget
oversampling along the readout direction is removed, e.g. 256 samples can be cut to 128
IsmrmrdReader
de-serialize the incoming ISMRMRD raw data package
Raw data
MRI scanner
Images
IsmrmrdImageWriterReader
send images back to scanner
AccumulatorGadget
accumulate the readout data and trigger the next gadget
GtPlusRecon2DTGadgetCloud
the main gadget to control the reconstruction
AutoScalingGadget
either apply a fixed scaling factor or automatically scale the images
ImageFinishGadget
attach related information, e.g. slice geometry etc., to images
TCP/IP connection
CloudConnector
CloudJobReader
de-serialize the cloud job data package
CloudJobWriter
send back the reconstructed results
GtPlusReconJob2DTGadget
first layer gadget
perform the computation of 2D+t reconstruction
TCP/IP connection
CloudConnector
CloudJobReader
CloudJobWriter
GtPlusReconJob2DTGadget
second layer gadget
CloudConnector
CloudJobReader
CloudJobWriter
GtPlusReconJob2DTGadget
second layer gadget

**FIG 4.**

The gadget chain used in the multi-slice myocardial cine imaging. Here the GtPlusRecon2DTGadgetCloud gadget controls the connection to the first layer node where the data from a slice was reconstructed. The second layer sub-nodes were further utilized to speed up the reconstruction. The TCP/IP socket connections were established and managed by the cloud software module implemented in the GT-Plus. The dotted line indicates the inter-node connection and the solid line means the data flow within the gateway and computing nodes.

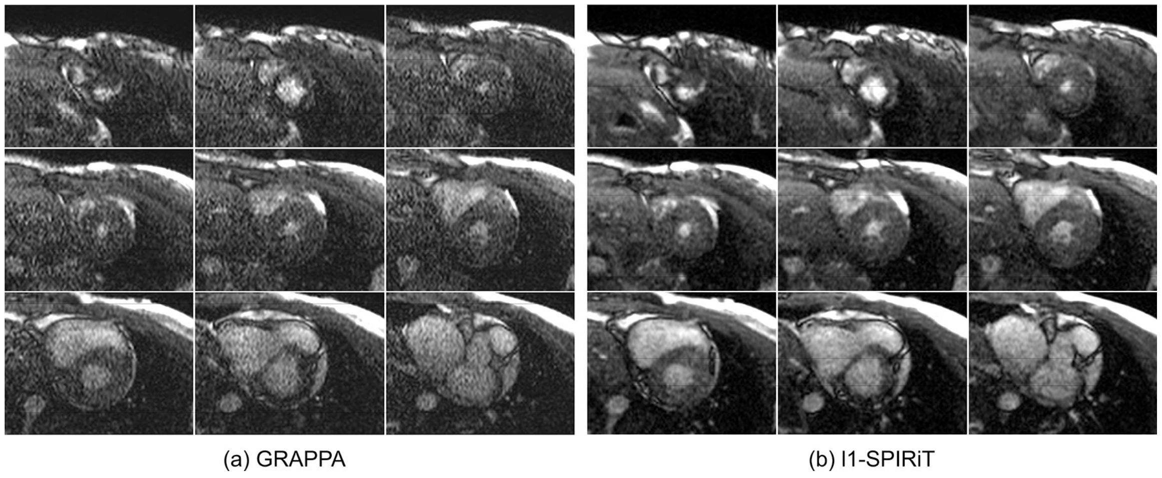


**FIG 5.**

Reconstruction results of multi-slice myocardial cine imaging on the GT-Plus cloud. Compared to the linear reconstruction (a), non-linear reconstruction (b) improves the image quality. With the computational power of Gadgetron based cloud, the entire imaging including data acquisition and non-linear reconstruction can be completed in 1min to achieve the whole heart coverage.

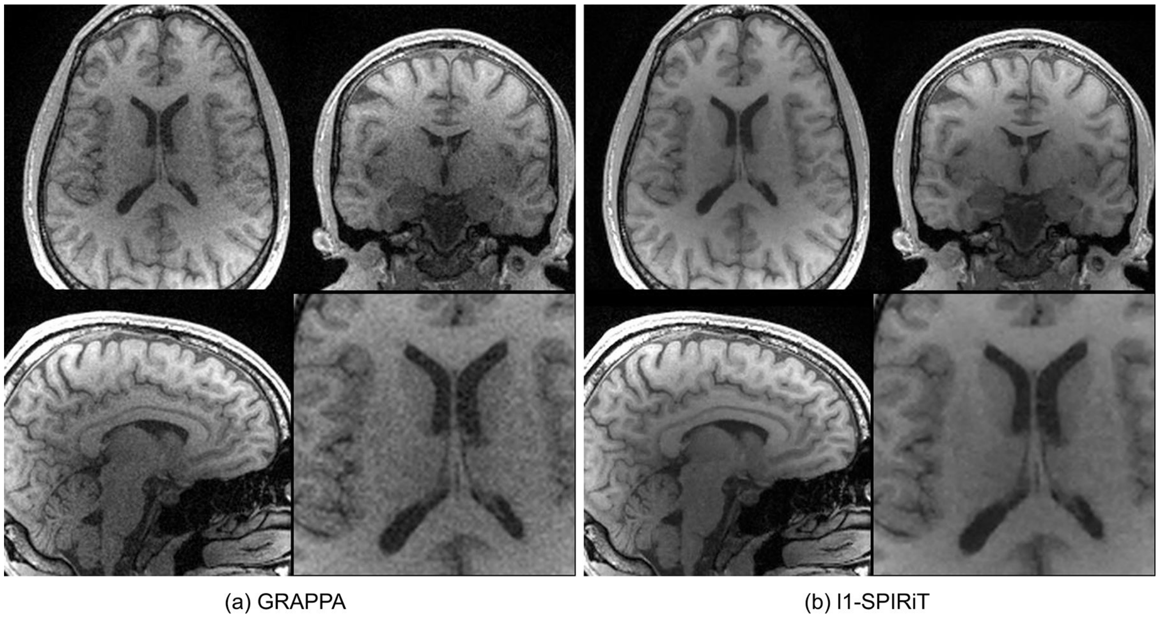


**FIG 6.**
Reconstruction results of 1mm$^3$ brain acquisition on the GT-Plus cloud. The basic cloud topology shown in Fig. 2 was used here. Both GRAPPA linear reconstruction (a) and (the l1SPIRiT results (b) are shown here. The single node processing time is over 20mins, which prohibits the clinical usage of non-linear reconstruction. With the Gadgetron based cloud, a computing time of <2.5mins can be achieved for this high resolution acquisition.

## Table 1

Total imaging time (from the start of data acquisition to the moment when all images are returned to the scanner) and computing time in seconds for the multi-slice myocardial cine imaging. The in vivo test was only performed with cloud setups. The single node computing time was recorded with the retrospective reconstruction on the gate-way node.

| | **Casual** | | **Biowulf** | | **Amazon EC2** | |
|---|---|---|---|---|---|---|
| | **Single** | **Cloud** | **Single** | **Cloud** | **Single** | **Cloud** |
| Imaging time (s) | - | 259.2 | - | 62.1 | - | 52.6 |
| Computing time (s) | 823.7 | 255.0 | 558.1 | 58.2 | 427.9 | 48.9 |

## Table 2

Total imaging and computing time in seconds for the 3D neuro acquisition. The single node used for comparison is the gateway node for every cloud setup.

| | **Casual** | | **Biowulf** | | **Amazon EC2** | |
|---|---|---|---|---|---|---|
| | **Single** | **Cloud** | **Single** | **Cloud** | **Single** | **Cloud** |
| Imaging time (s) | - | 541.9 | - | 278.5 | - | 255.0 |
| Computing time (s) | 1054.3 | 449.1 | 1265.1 | 186.4 | 983.5 | 146.0 |