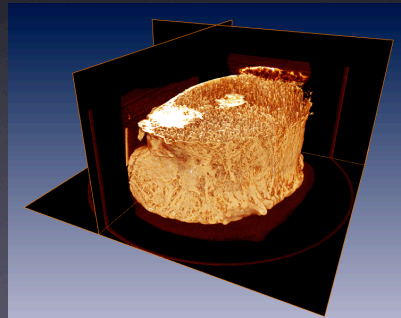


Interactive Exploration of Large Remote μ -CT Scans

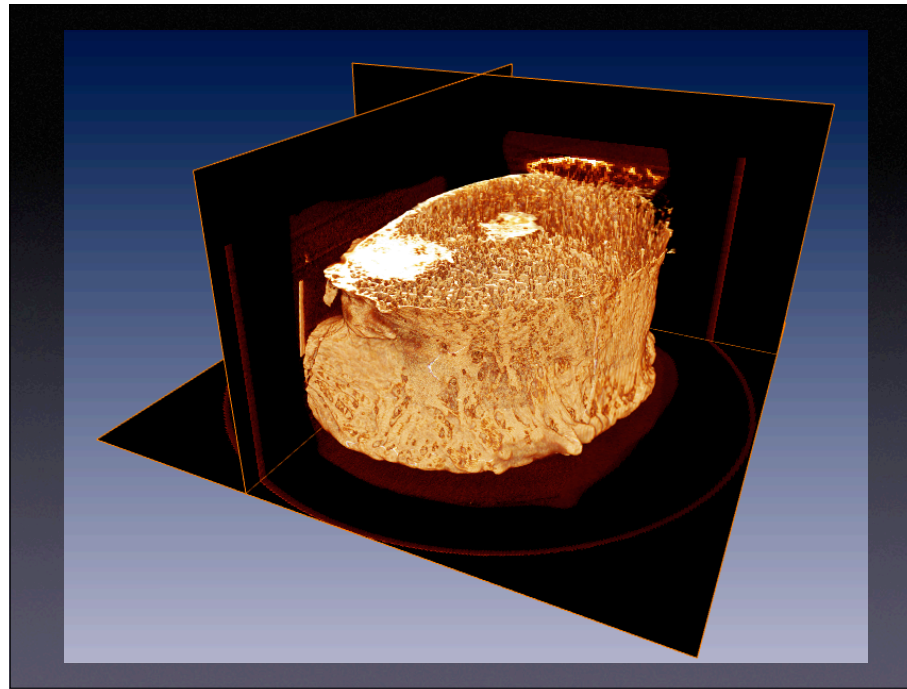
Steffen Prohaska • Andrei Hutanu
Ralf Kähler • Hans-Christian Hege
Zuse Institute Berlin



Thanks for the introduction.

In the lower part of the slide you can see volume renderings of human vertebral bodies. **[PRESS]** Vertebral bodies are the bones that make up our spine. Please appreciate the rich internal structure. **[PRESS]** This is the trabecular bone which is the calcified tissue playing an important role for the stability of the bone. During bone diseases, such as Osteoporosis, this structure deteriorates. A similar process takes place during space flights of humans. This is the reason, why Space Agencies like ESA and NASA are interested in a better understanding of the underlying mechanisms.

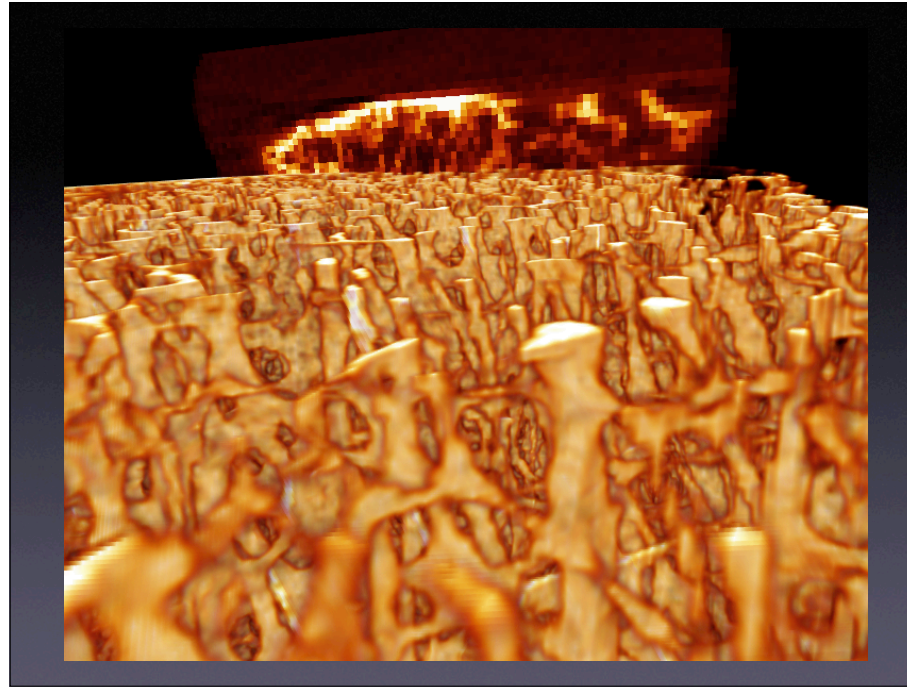
A research group which is spread over Europe **[NEXT SLIDE]** uses the system I'm presenting in my talk in bone research.



Thanks for the introduction.

In the lower part of the slide you can see volume renderings of human vertebral bodies. **[PRESS]** Vertebral bodies are the bones that make up our spine. Please appreciate the rich internal structure. **[PRESS]** This is the trabecular bone which is the calcified tissue playing an important role for the stability of the bone. During bone diseases, such as Osteoporosis, this structure deteriorates. A similar process takes place during space flights of humans. This is the reason, why Space Agencies like ESA and NASA are interested in a better understanding of the underlying mechanisms.

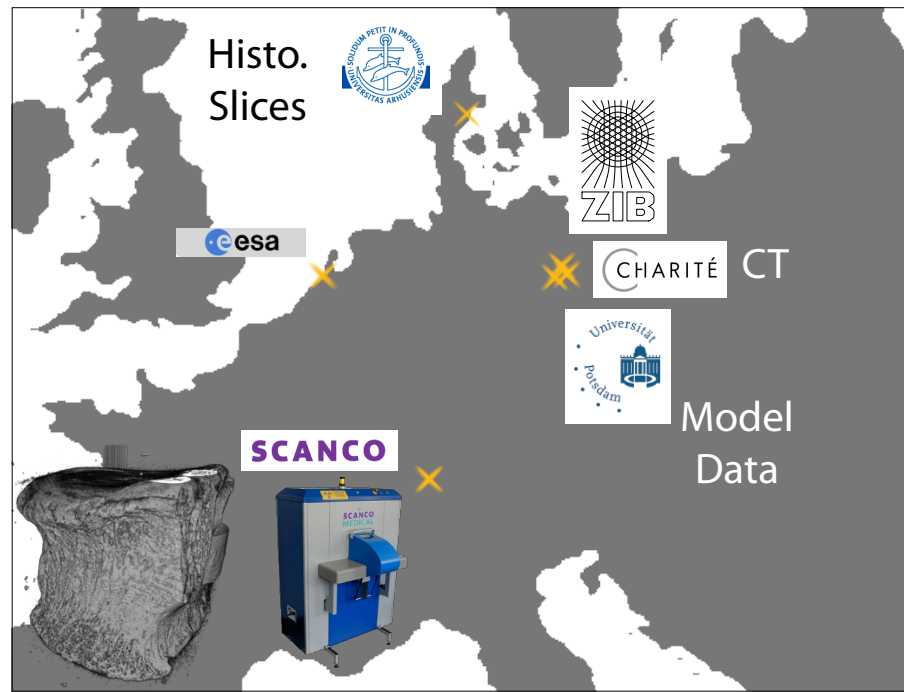
A research group which is spread over Europe **[NEXT SLIDE]** uses the system I'm presenting in my talk in bone research.



Thanks for the introduction.

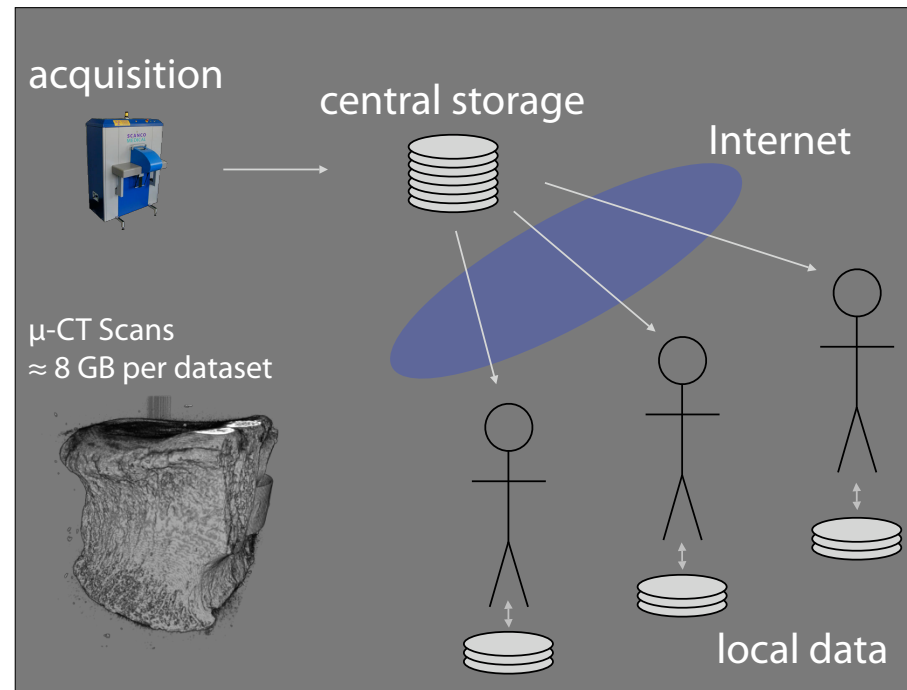
In the lower part of the slide you can see volume renderings of human vertebral bodies. **[PRESS]** Vertebral bodies are the bones that make up our spine. Please appreciate the rich internal structure. **[PRESS]** This is the trabecular bone which is the calcified tissue playing an important role for the stability of the bone. During bone diseases, such as Osteoporosis, this structure deteriorates. A similar process takes place during space flights of humans. This is the reason, why Space Agencies like ESA and NASA are interested in a better understanding of the underlying mechanisms.

A research group which is spread over Europe **[NEXT SLIDE]** uses the system I'm presenting in my talk in bone research.



A research group which is spread over Europe [**cont.**] uses the system I'm presenting in my talk in bone research.

Data from different sources is used: Data from Standard CT scanners, data computed by a bone structure model and histological slices acquired by cutting the bone is among them. This data is acquired locally at the participating institutions. But by far the largest datasets come from a high resolution micro CT Scanner. Complete bones are scanned at a resolution of 40μ . The image size is $2k \times 2k$ by one to two thousand slices at 12bit resolution—Resulting in about 8GB per volume—And about 70 of such volumes are used in a study. The goal is to learn about processes changing the trabecular bone and to develop measures to monitor these. The interesting features in the data are not yet known in detail. So being able to explore the data is important. The sheer size of the micro CT scans make them difficult to handle. [**NEXT SLIDE**] We decided to handle them...



[cont.] We decided to store them [the micro CT scans] centrally. Hardware resources at the institutions were limited. Therefore we tried to use standard PC hardware which was already available. A fast Internet connection is available at all sites.

Our system is able to remotely access the micro CT scans and provide standard visualization techniques. Local data can be loaded at the same time. Storing subblocks of the remote data for later processing is also possible.

I'll show some of the features in a short movie **[NEXT SLIDE]**



[cont.] This is a live capture showing our major visualization techniques. Data is accessed remotely. All rendering is done using the local graphics hardware.

An extended version of the movie is available at

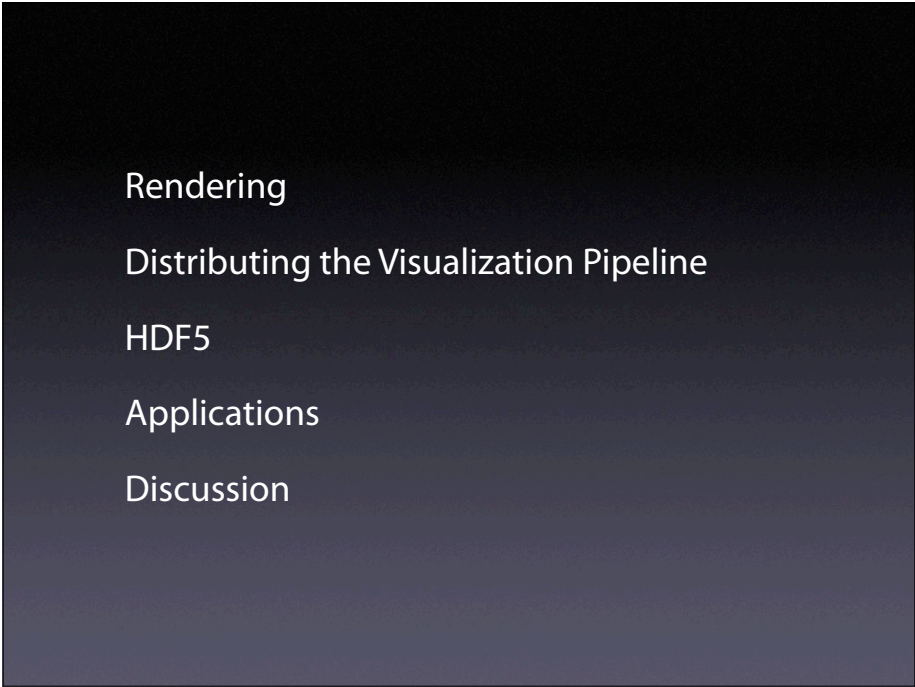
http://www.zib.de/prohaska/2004/movies/Prohaska_et_al_Vis2004.avi

First you see a slice through a micro CT data sets. It progressively refines the resolution of the data after it was moved to a new location.

In the next part a point of interest is selected and a volume renderer is switched on. It uses a hierarchical representation of the data with a higher resolution around the point of interest. Data is retrieved in the background allowing user interaction at all times.

Now the user changes the point of interest. The volume renderer adjusts to this and retrieves high resolution data at the new location.

I'll now give an outline of the remaining talk **[NEXT SLIDE]**



- Rendering

- Distributing the Visualization Pipeline

- HDF5

- Applications

- Discussion

[cont.] First I'll briefly cover our rendering techniques.

Followed by a few words on how we distributed the visualization pipeline and a more in depth discussion about how we use HDF5 for data access.

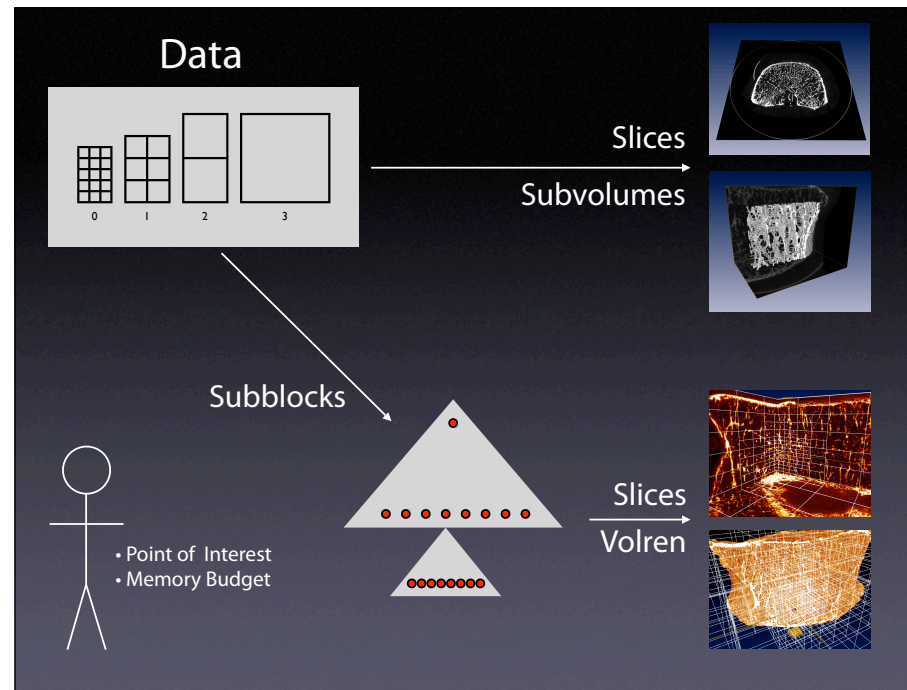
After showing some applications of our system, I'll conclude with discussing results.

[NEXT SLIDE]

Let's start with our rendering algorithms

Rendering

[cont.] Let's start with our rendering algorithms **[NEXT SLIDE]**



[cont.] The volumetric data is stored at full resolution. Additionally, a hierarchy of lower resolution preview versions is generated. To preserve thin structures in our data we use a maximum filter in the first downsampling step. Further levels are generated by averaging the voxel values of the level just below.

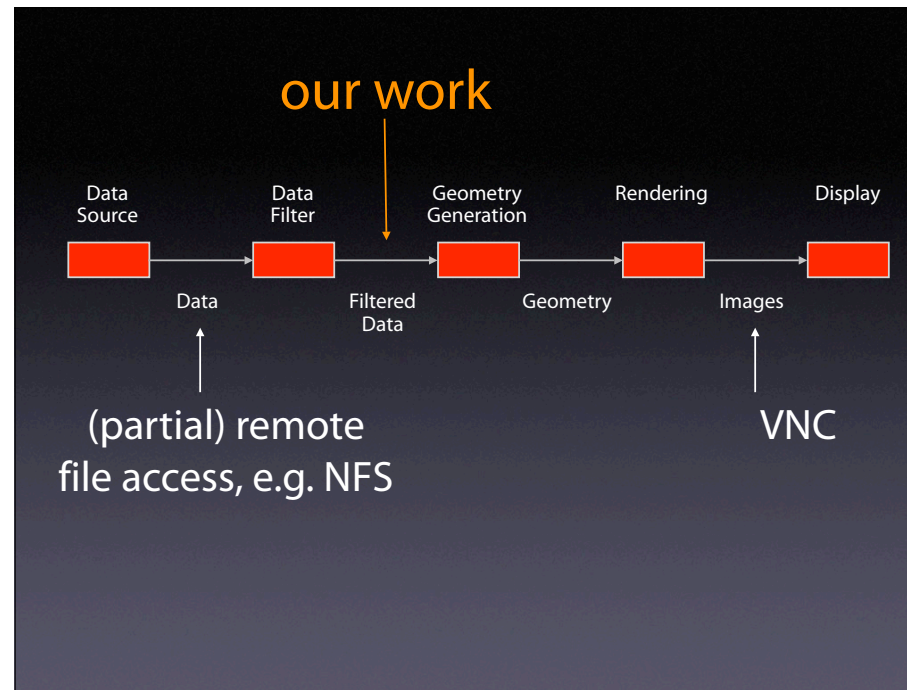
[PRESS] Accessing subvolumes of a specific resolution is used for simple slicing or for retrieving subvolumes to be stored locally or be visualized using various techniques.

[PRESS] Our hierarchical rendering methods use an octree representation of the data. A user specified Point of Interest and a Memory Budget control the filling of the octree **[PRESS]** which is then accessed for slicing and volume rendering. We use a standard hardware volume rendering approach which traverses the octree to build up textures for nodes without children.

All data retrieval is performed in a background thread. The rendering is updated when new data becomes available. Implementation of the system is rather straight forward if data is stored on a local disk. **[NEXT SLIDE]** The next step is to distribute the viz pipeline.

Distributing the Visualization Pipeline

[cont.] The next step is to distribute the visualization pipeline. **[NEXT SLIDE]**



[cont.] We considered methods transferring the rendered images. Remote desktop solutions like VNC could be used with nearly no development effort. Besides potential network latency problems we think the major restriction is missing access to data stored at more than one location to generate a single visualization. Data stored local on the machine displaying the images had to be transferred to the rendering server first.

Remote file access as the other extrem is also very simple. Security considerations might make this solution impractical. But more severely it suffers from network latency which I'll discuss in more depth shortly.

Our solution uses a server running on a machine with local access to the full datasets. It extracts subvolumes of the data which are send to the client.

[NEXT SLIDE] Our solution is based on HDF5

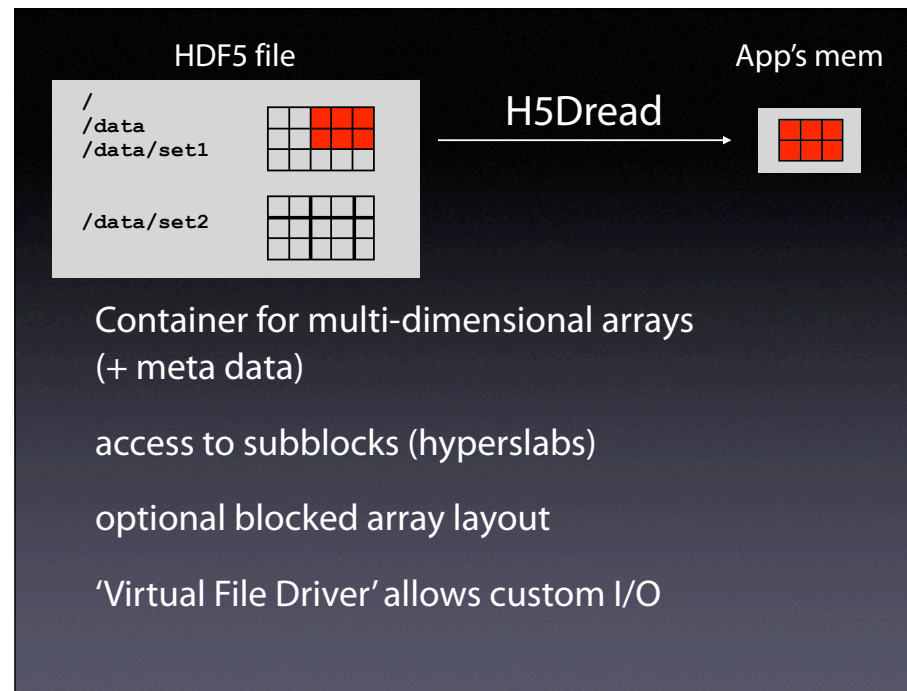
<http://hdf.ncsa.uiuc.edu/HDF5>

HDF5

[cont.] Our solution is based on HDF5.

HDF5 is a file format for scientific data together with tools and a developer API to access the data. It is developed at NCSA and available as an open source library. You can download it at the url at the top.

[NEXT SLIDE] It provides a container for multidimensional arrays...

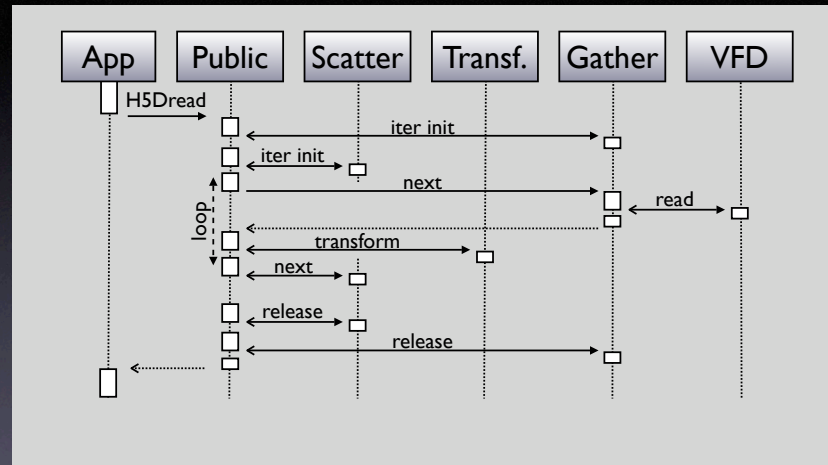


[cont.] It provides a container for multidimensional arrays. Several of these arrays can be stored together with accompanying meta data. They can be organized in a file system like structure.

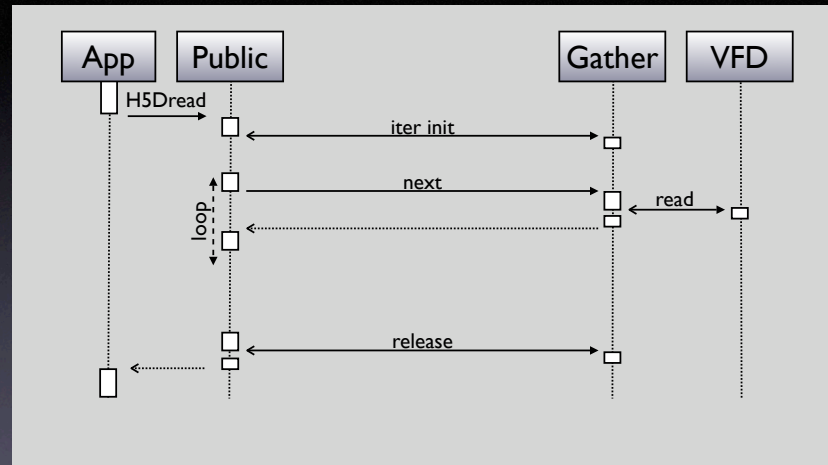
The API provides functions to access subvolumes of a dataset. The API is very suitable for volume visualization. Blocked array layouts can be used to optimize retrieval of subblocks by increasing data locality in the file. Compression is also available transparently inside the HDF5 library.

A Virtual File Driver layer in the library allows to implement custom I/O methods. This could be used to implement remote access. But a straight forward implementation based on partial remote file access suffers from performance problems.

These get clear after a look into the internals of a read operation **[NEXT SLIDE]**

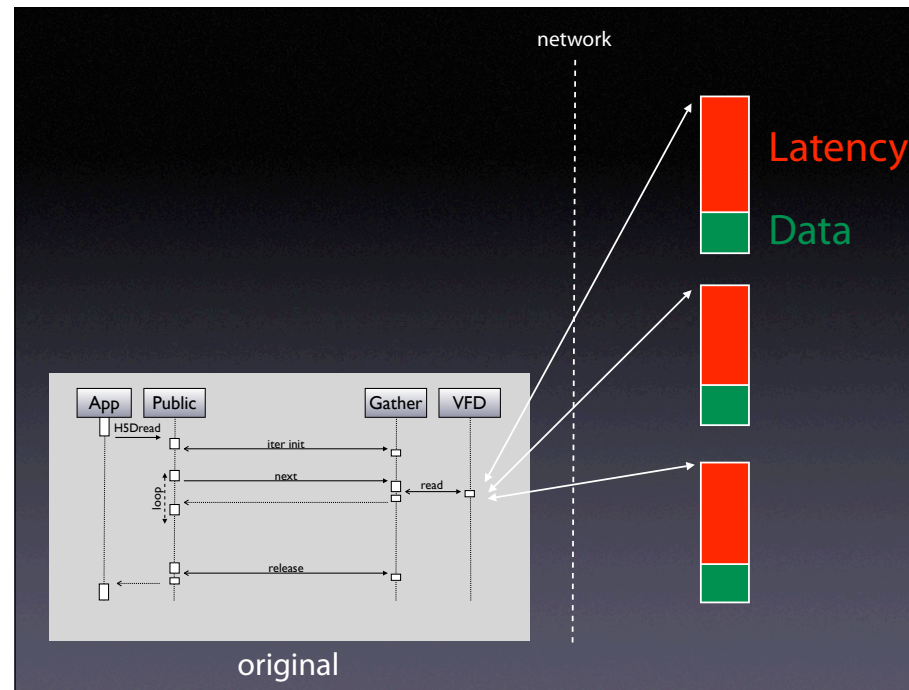


[cont.] This is a high level sequence diagram of a read operation inside the HDF5 Library. It's basically built up of two iterators. One is gathering data from a file, the second one is writing the data to the application's memory. Only the read operations are of interest for understanding the remote performance. **[NEXT SLIDE]**



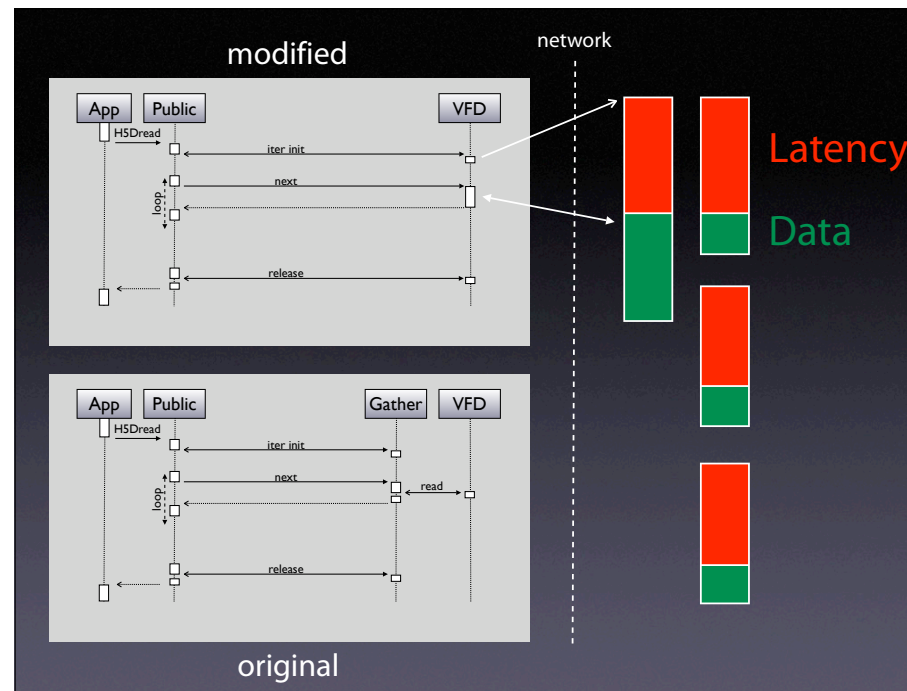
[cont.] After the application requests a certain subvolume, an iterator walks over this selection and generate read operations to bring in portions of the file needed to serve the request. These read operations are sent to the virtual file driver, depicted on the very right. Locally these operation are just sent to the file system.

For remote access they must be sent to the remote side. **[NEXT SLIDE]**



[cont.] The network introduces a lot more latency than a local file system. One request for a subvolume might generate a large amount of read operations to the virtual file driver. Each adds latency and stalls the hdf5 read operation. Depending on the quality of the network this has a major impact on the overall performance. It might decrease by an order of magnitude compared to a local operation. Bandwidth of the network is not the limiting factor. Only a small percentage is used. In the original HDF5 implementation this problem could not be solved.

Therefore, we modified the HDF5 internals. **[NEXT SLIDE]** We changed how a virtual file driver is accessed.

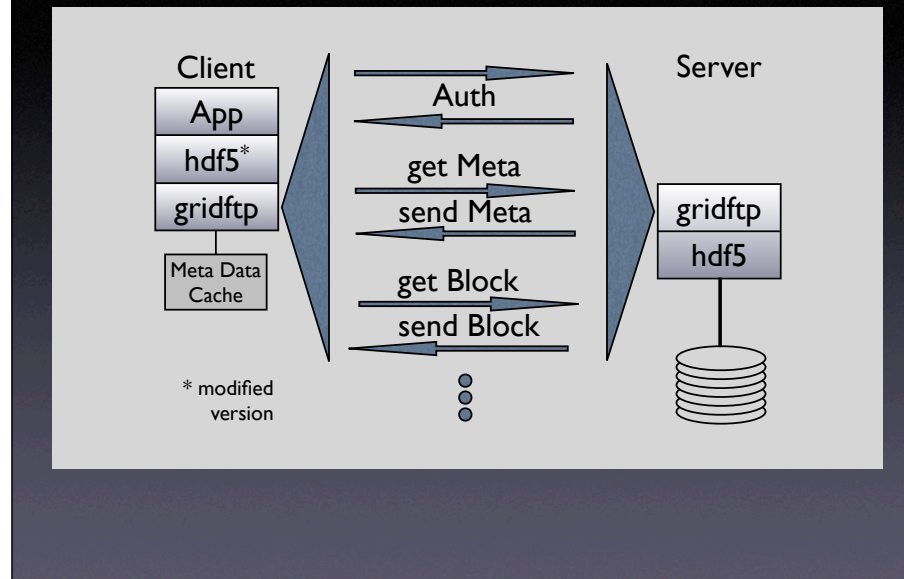


[cont.] We changed how a virtual file driver is accessed. Instead of sending unrelated read requests it is given the opportunity to see the whole selection. During iterator initialization it sends this request to the remote side. Processing there starts right away. Data is sent back and delivered to the HDF5 library on demand.

Using this modification only one read request is sent to the remote side per subvolume request by the application. Depending on the block size requested by the application we achieve about 10–20% usage of the bandwidth of a 100 Mbit Ethernet.

A suitable protocol must be used to request the selection from the remote side. **[NEXT SLIDE]** We chose to use gridftp which is an extension to the standard FTP protocol.

<http://www.zib.de/visual/projects/gridlab/hdf5/>



[cont.] We chose to use gridftp which is an extension to the standard FTP protocol. It provides server side processing information. The overall architecture of the remote access looks like this. All our modifications to existing software are available at the url at the top.

On the client side, the application uses HDF5 with modified internals. GridFTP is used to implement the network I/O. It communicates with a GridFTP server which uses HDF5 to access the data file. From the application's view the interface to the data is the same for local or remote access.

To optimize a remote file open we send all HDF5 meta data in one block after authentication and store them in a meta data cache locally. Later, only sub block requests are sent to the remote side.

I'm now going to present some applications of our system **[NEXT SLIDE]**

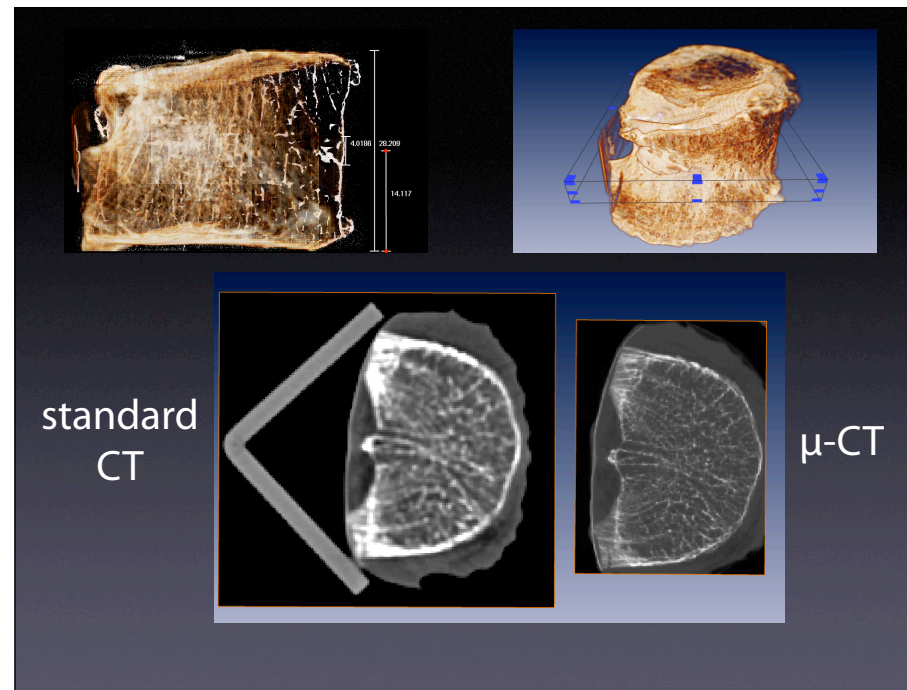
Applications

[cont.] As mentioned before, it is not yet clear which the relevant features in the dataset are. Our software is mainly used to explore the data and to find matching volumes in different images of the same bone.

These comparisons are used to validate imaging technologies and to select similar volumes for further data analysis.

Subblocks of the data are also use to test new evaluation procedures.

[NEXT SLIDE] In the first example data from a clinical CT scanner and the data acquired using a micro-CT is matched.



[NEXT SLIDE] In the first example data from a clinical CT scanner and the data acquired using a micro-CT is matched.

The location and geometry of the conventional CT slices is approximately known. Therefore it is easy to select a central slice of the right thickness. This is depicted at the top. The selection is then average it in z direction and visually compared to a slice from the clinical CT scanner (bottom).

This is used to ‘simulate’ a high resolution CT slice which could not be acquired otherwise.

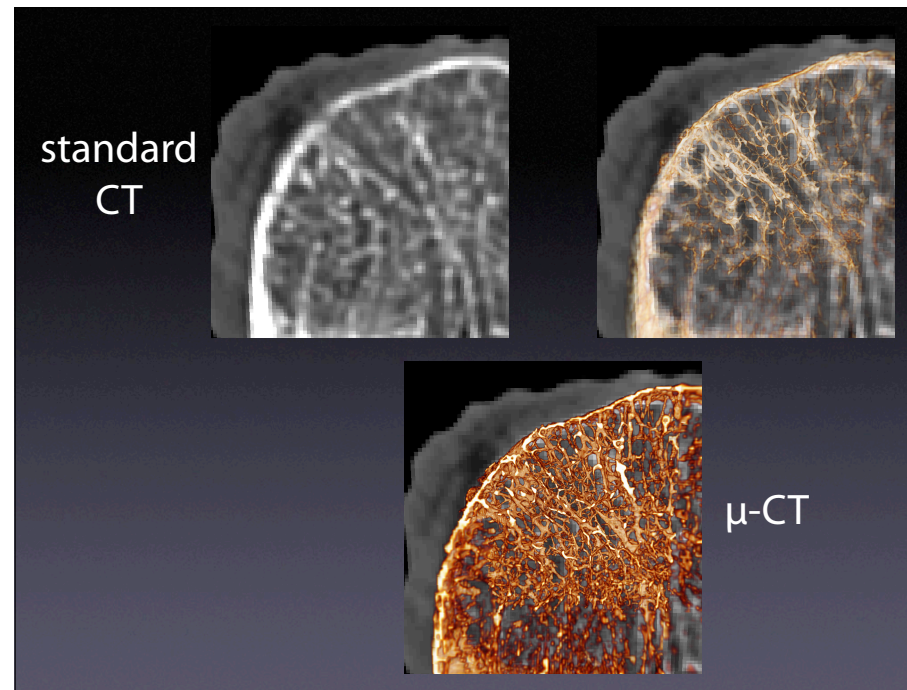
[NEXT SLIDE] Analyzing the same region in the slice and in the 3D volume is also possible.

Top left: Select central 4mm slice

Bottom Left: CT scan

Top right: 3D view of selected volume

Bottom Right: micro-CT averaged in z direction

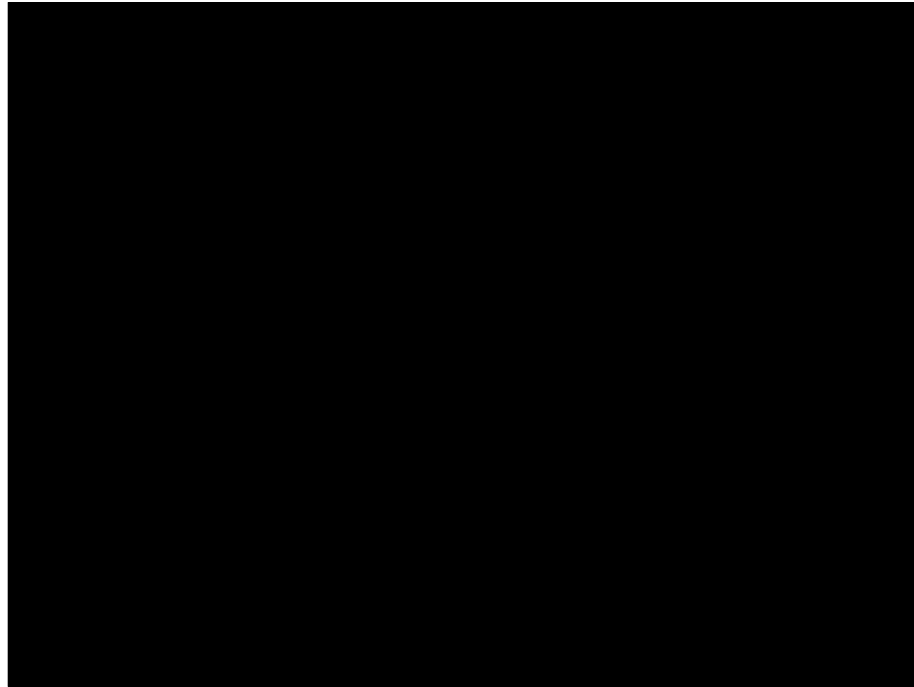


[cont.] Analyzing the same region in the slice and in the 3D volume is also possible.

Here a smaller area of the CT slice together with a volume rendering of the micro-CT data is shown.

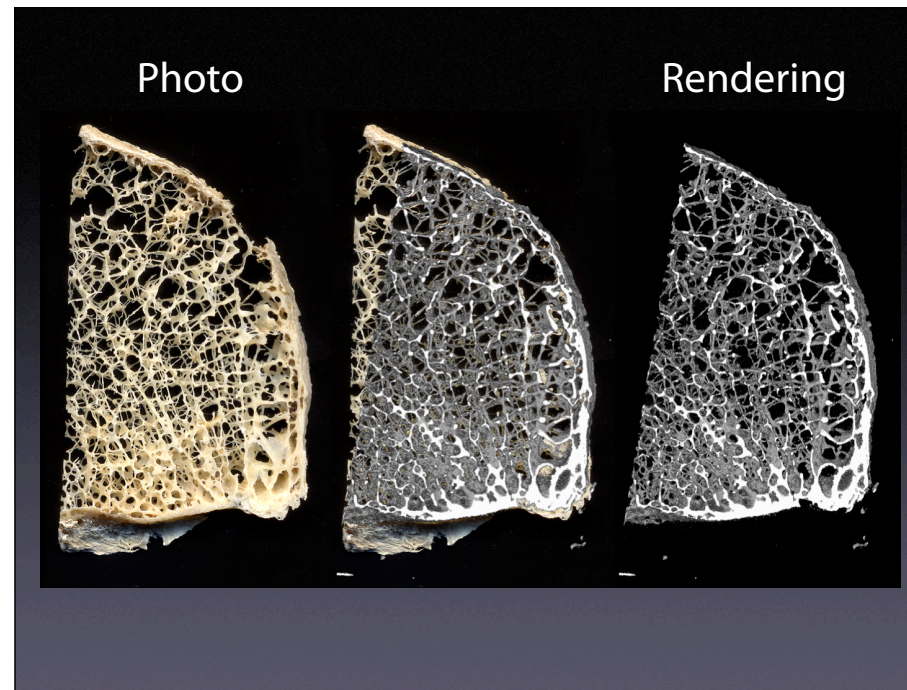
Finding exact matches of data from different sources facilitates evaluation and understanding of the measures used to quantify the image data.

[NEXT SLIDE]



[cont.] In another part of the project the bones are cut into histological slices which are analyzed afterwards. Slices with a thickness of 10μ are cut from a bone which was embedded in plastic. Although this technique requires a lot of processing and introduces artifacts into the structure of the bone, measures based on these technique are still the 'gold standard' for assessing bone structure.

[NEXT SLIDE] Finding the histological slices in the micro CT data could be useful for validation. We are not yet able to do this for thin slices. It's much easier for a thicker slice which is shown here.



[cont.] Finding the histological slices in the micro CT data could be useful for validation. We are not yet able to do this for single slices. It's much easier for a thicker slice which is shown here.

The left side shows a photograph taken of a thick slice cut out of a vertebral body. On the right hand side a subvolume of a micro CT scan is volume rendered. These volumes were manually matched.

With this last example I'll conclude the applications and will move on. **[NEXT SLIDE]**

Discussion

[cont.] I'll briefly wrap up [NEXT SLIDE]

Rendering

Hierarchical methods to provide overview and details at the same time

Wavelet methods could be used to reduce amount of transferred data/improve image quality—They would fit into our framework

[cont.] We are using basic hierarchical methods for rendering. Our methods are not optimized to reduce the amount of transferred data. Due to fast internet connections at all sites, this was not a problem. Getting a useful system working was our first concern.

Wavelet methods could be used to reduce the amount of transferred data. Our preview generation could be replaced by a wavelet scheme. Our remote access methods would still be applicable. **[NEXT SLIDE]**

Data Access Using HDF5

- + Stable API and file format
- Restricts optimization

Improved remote performance by reducing network latency

Open issues

- We 'switched off' HDF5 caches
- No asynchronous API
- No API to cancel operations
- Only one thread in the library at the same time

[cont.] We started with a lot of enthusiasm into using HDF5 as our file format. It provides a stable API and file format and is widely used. But it restricts potential optimization when building an application.

We improved the remote performance by reducing network latency. This was done without modifying the application's view on HDF5. If HDF5 provides another way of optimizing remote access in the future, we will get it for free. But our changes switch off HDF5 caches. This is not harmful in our case, because our octree data structure itself is a data cache.

[NEXT SLIDE] HDF5 also has some other issues.

Data Access Using HDF5

- + Stable API and file format
- Restricts optimization

Improved remote performance by reducing network latency

Open issues

- We 'switched off' HDF5 caches
- No asynchronous API
- No API to cancel operations
- Only one thread in the library at the same time

[cont.] HDF5 also have some other issues.

It does not provide an asynchronous API. This forced us to use a background thread to fetch data while maintaining interactive rendering. If HDF5 directly supported asynchronous access our application could be simpler.

An API to cancel operations is also missing. Together with the fact, that only one thread may be in the library at the same time, this might cause blocking behaviour. If a user changes the point of interest or the location of a slice, we have to wait until the running read operation finishes, before we can adjust to these changes.

[NEXT SLIDE] to finally summarize

Final Points

(improved) HDF5 can be used for efficient remote data access

Responsiveness might be a problem

What's 'the' best (standard) file format and API for out-of-core and remote access to hierarchical structured grids for interactive and responsive visualization?

[cont.] HDF5 can be used for efficient remote data access

It has some issues when used in an interactive and responsive environment.

So for me, the question remains, what's the best file format and API for out-of-core and remote access to hierarchical structured grids for interactive and responsive visualization?

That's it from my side, Thank you.

