CIF21 DIBBs: Ubiquitous Access to Transient Data and Preliminary Results via the SeedMe Platform

Award Number : 1443083
PI: Amit Chourasia, Co-PI: Michael Norman
San Diego Supercomputer Center, University of California San Diego

Open source ✓
Web based ✓
Cross-platform ✓

What is SeedMe?

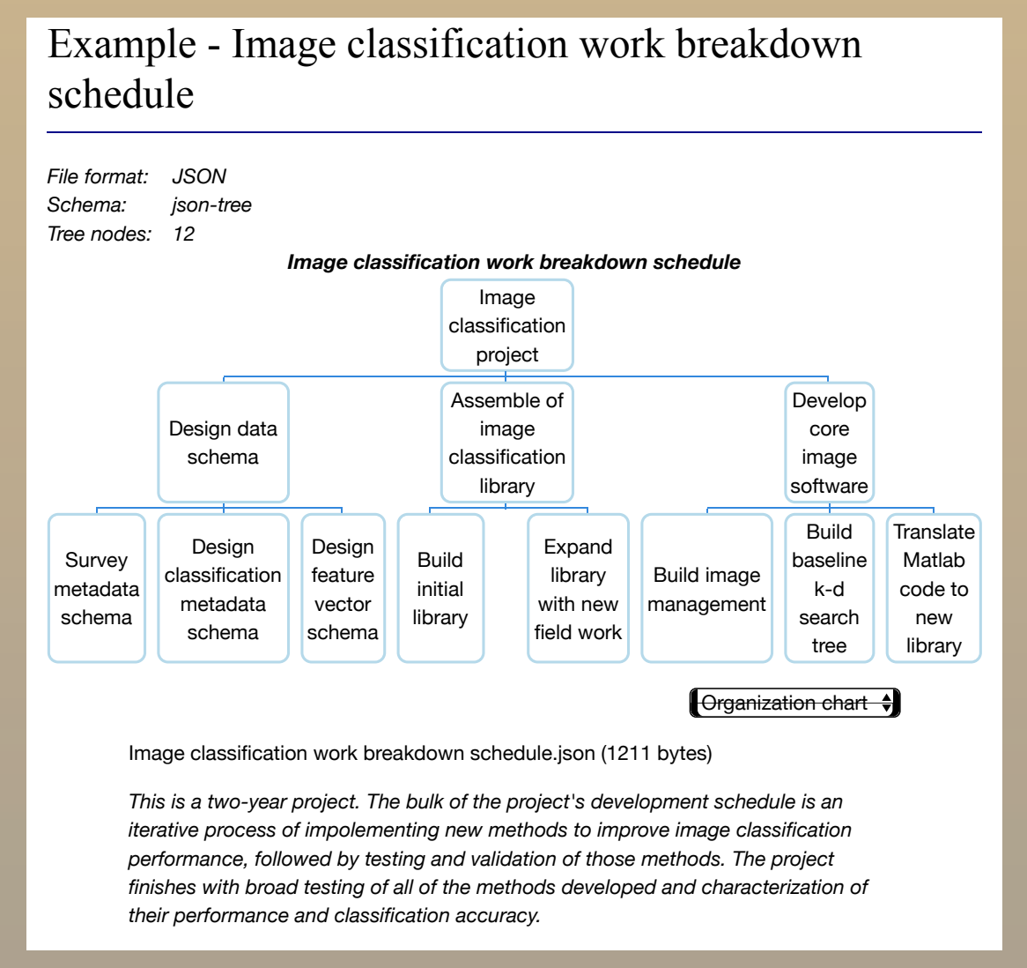
SeedMe = Stream Encode Explore and Disseminate My Experiments

SeedMe is a platform that enables easy sharing of transient and preliminary data for a broad research computing community by offering cyberinfrastructure as a service and a modular software stack that may be customized. SeedMe is based on Drupal content management system with a set of building blocks with additional PHP modules and web services clients.

Planned building blocks

Federated authentication + Authorization	Virtual File Sytem	Access Control
Sharing	Search / Index	Microformats
Field Formatters	REST API	Clients (Java, Python) + Command Line
Light Visualization	Rich Text	Discussion

Light visualizations



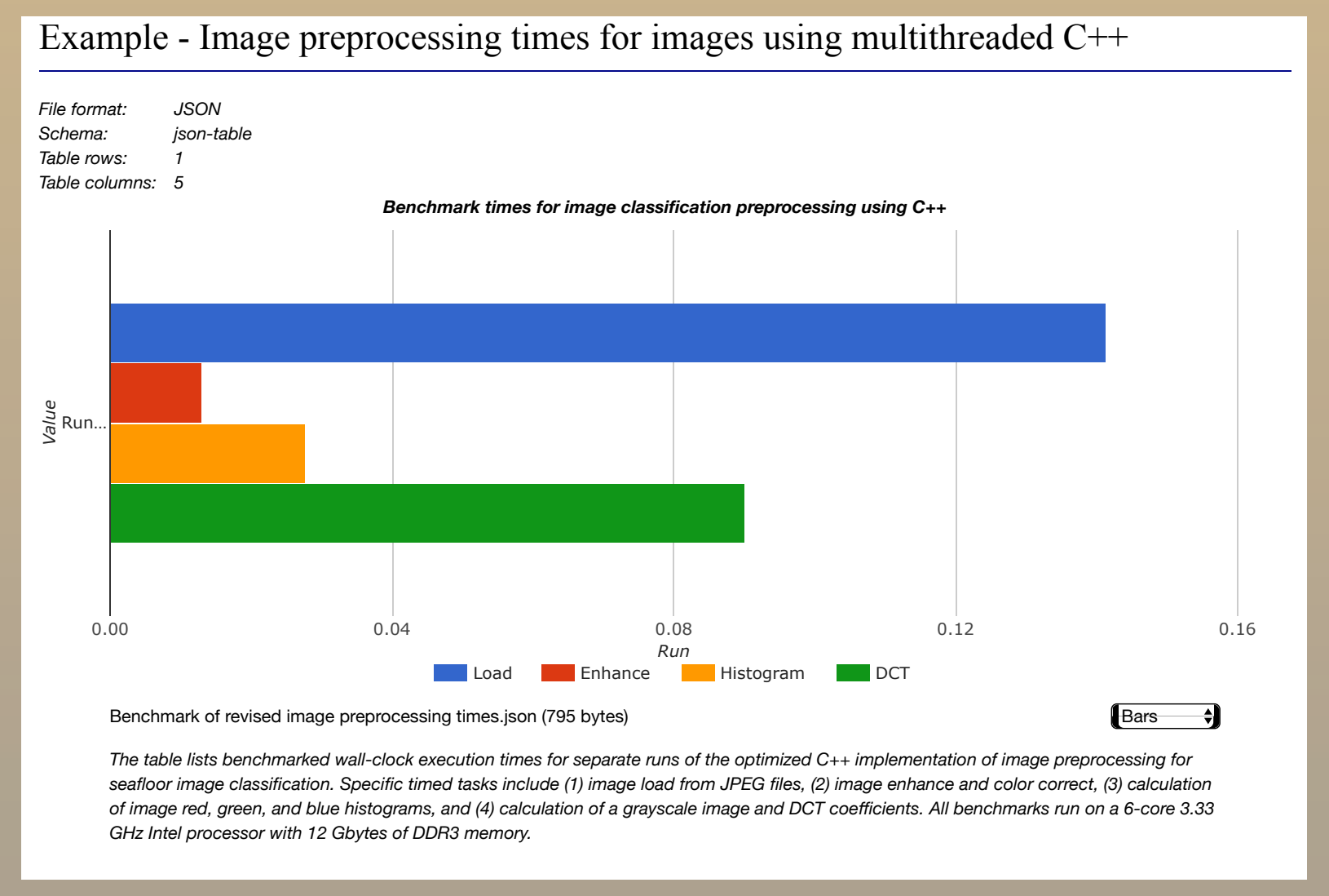
Example - Image classification work breakdown schedule

File format: *JSON*
Schema: *json-tree*
Tree nodes: *12*

Image classification work breakdown schedule
Image classification project
Design data schema
Assemble of image classification library
Develop core image software
Survey metadata schema
Design classification metadata schema
Design feature vector schema
Build initial library
Expand library with new field work
Build image management
Build baseline k-d search tree
Translate Matlab code to new library
Organization chart

Image classification work breakdown schedule.json (1211 bytes)

This is a two-year project. The bulk of the project's development schedule is an iterative process of impolementing new methods to improve image classification performance, followed by testing and validation of those methods. The project finishes with broad testing of all of the methods developed and characterization of their performance and classification accuracy.



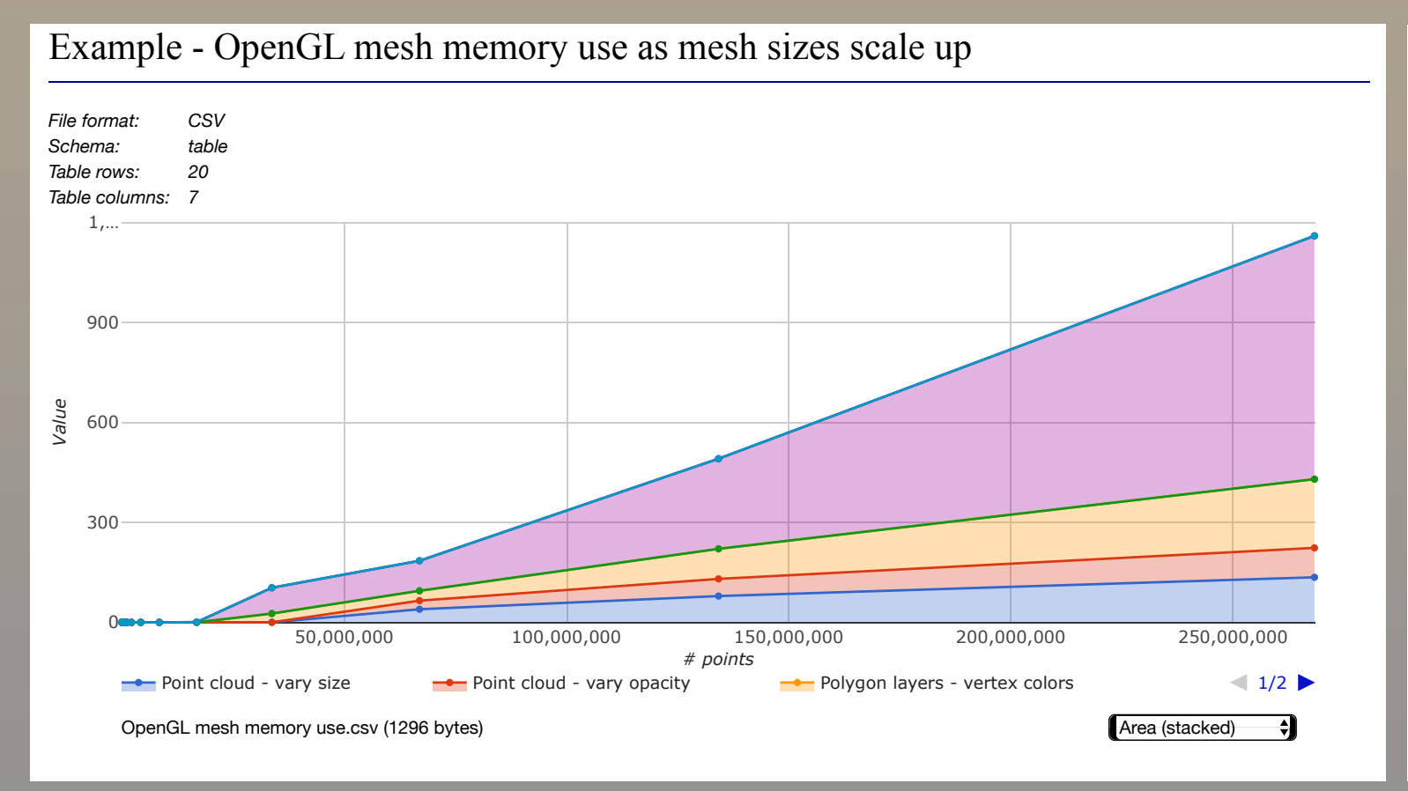
Example - Image preprocessing times for images using multithreaded C++

File format: *JSON*
Schema: *json-table*
Table rows: *1*
Table columns: *5*

Benchmark times for image classification preprocessing using C++
Value
Run...
0.00
0.04
0.08
0.12
0.16
Run
Load
Enhance
Histogram
DCT
Bars

Benchmark of revised image preprocessing times.json (795 bytes)

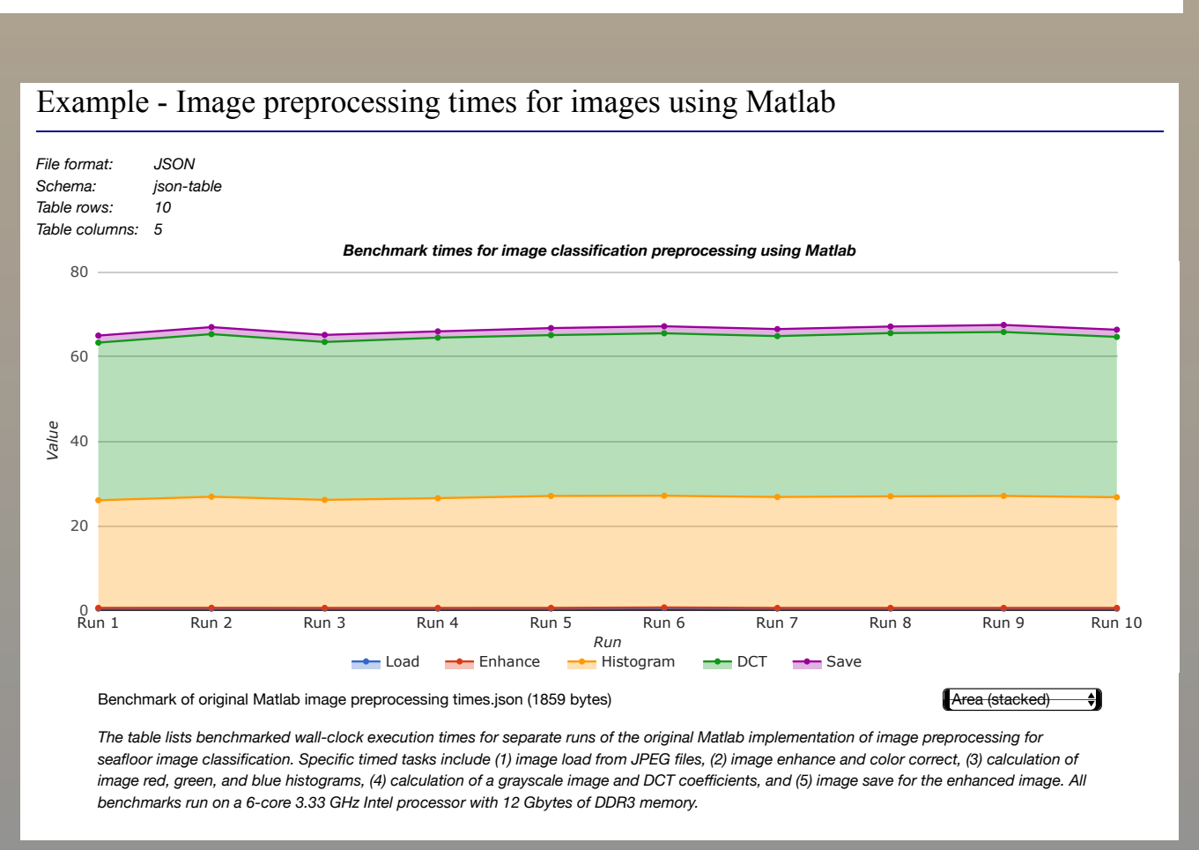
The table lists benchmarked wall-clock execution times for separate runs of the optimized C++ implementation of image preprocessing for seafloor image classification. Specific timed tasks include (1) image load from JPEG files, (2) image enhance and color correct, (3) calculation of image red, green, and blue histograms, and (4) calculation of a grayscale image and DCT coefficients. All benchmarks run on a 6-core 3.33 GHz Intel processor with 12 Gbytes of DDR3 memory.



Example - OpenGL mesh memory use as mesh sizes scale up

File format: *CSV*
Schema: *table*
Table rows: *20*
Table columns: *7*

OpenGL mesh memory use.csv (1296 bytes)



Example - Image preprocessing times for images using Matlab

File format: *JSON*
Schema: *json-table*
Table rows: *10*
Table columns: *5*

Benchmark of original Matlab image preprocessing times.json (1859 bytes)

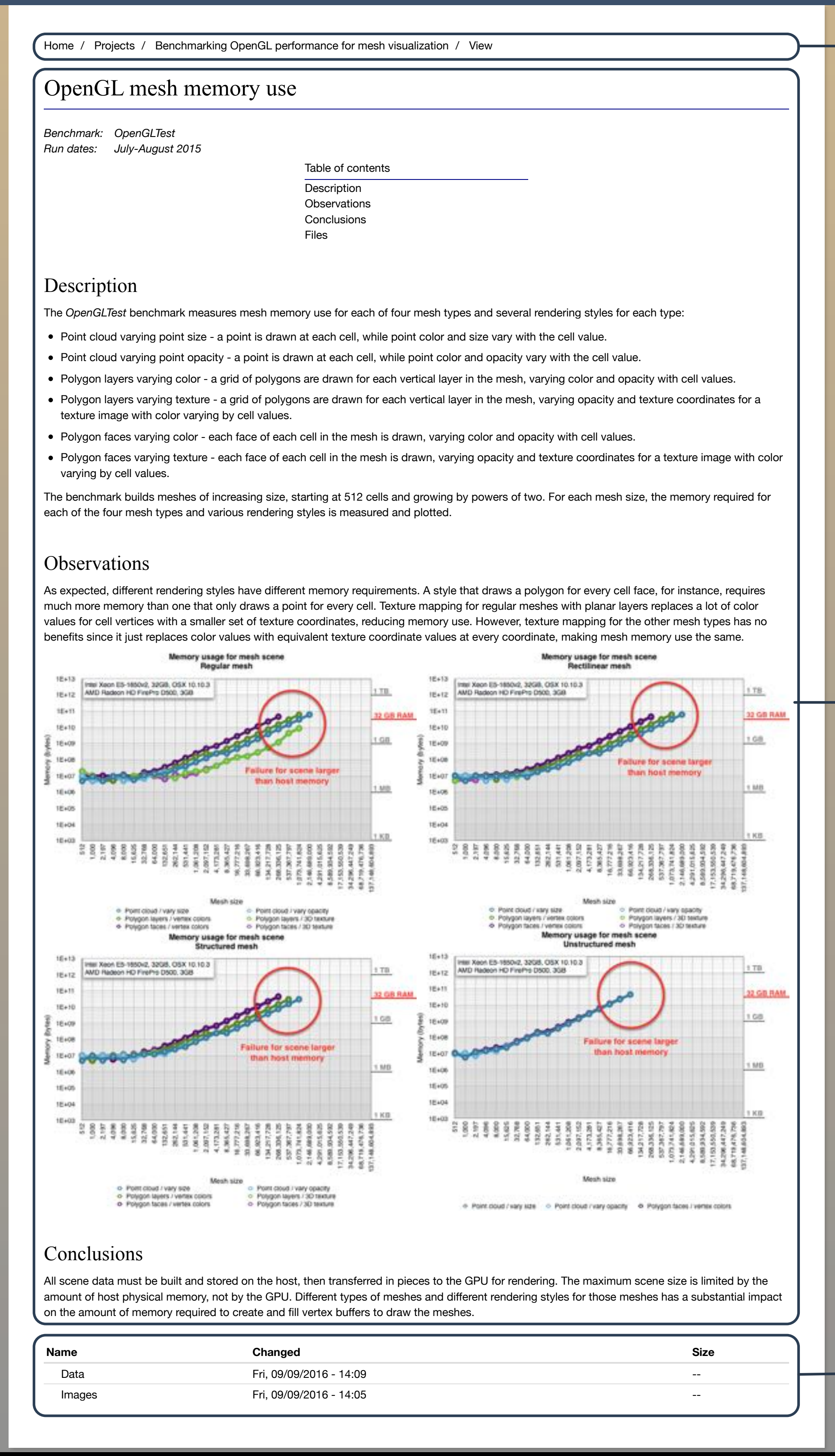
The table lists benchmarked wall-clock execution times for separate runs of the original Matlab implementation of image preprocessing for seafloor image classification. Specific timed tasks include (1) image load from JPEG files, (2) image enhance and color correct, (3) calculation of image red, green, and blue histograms, (4) calculation of a grayscale image and DCT coefficients, and (5) image save for the enhanced image. All benchmarks run on a 6-core 3.33 GHz Intel processor with 12 Gbytes of DDR3 memory.

Significant results

- ✓ Federated authentication
- Virtual File Sytem
- ✓ Small Data Visualization
- Small Data API ✓
- ✓ Pilot project with 580+ registered users
120,000+ content items
Integration with four scientific tools
- Early demonstration prototype ✓

✓ Available

Virtual file system: Sample UI



Home / Projects / Benchmarking OpenGL performance for mesh visualization / View

OpenGL mesh memory use

Benchmark: *OpenGLTest*
Run dates: *July-August 2015*

Table of contents
Description
Observations
Conclusions
Files

Description

The *OpenGLTest* benchmark measures mesh memory use for each of four mesh types and several rendering styles for each type:

- Point cloud varying point size - a point is drawn at each cell, while point color and size vary with the cell value.
- Point cloud varying point opacity - a point is drawn at each cell, while point color and opacity vary with the cell value.
- Polygon layers varying color - a grid of polygons are drawn for each vertical layer in the mesh, varying color and opacity with cell values.
- Polygon layers varying texture - a grid of polygons are drawn for each vertical layer in the mesh, varying opacity and texture coordinates for a texture image with color varying by cell values.
- Polygon faces varying color - each face of each cell in the mesh is drawn, varying color and opacity with cell values.
- Polygon faces varying texture - each face of each cell in the mesh is drawn, varying opacity and texture coordinates for a texture image with color varying by cell values.

The benchmark builds meshes of increasing size, starting at 512 cells and growing by powers of two. For each mesh size, the memory required for each of the four mesh types and various rendering styles is measured and plotted.

Observations

As expected, different rendering styles have different memory requirements. A style that draws a polygon for every cell face, for instance, requires much more memory than one that only draws a point for every cell. Texture mapping for regular meshes with planar layers replaces a lot of color values for cell vertices with a smaller set of texture coordinates, reducing memory use. However, texture mapping for the other mesh types has no benefits since it just replaces color values with equivalent texture coordinate values at every coordinate, making mesh memory use the same.

Memory usage for mesh scene
Regular mesh
Memory usage for mesh scene
Rectilinear mesh
Intel Xeon E5-1650v2, 32GB, OSX 10.10.3
AMD Radeon HD FirePro D500, 3GB
Memory (bytes)
1 TB
32 GB RAM
1 GB
1 MB
1 KB
Failure for scene larger than host memory
Mesh size
Point cloud / vary size
Point cloud / vary opacity
Polygon layers / vertex colors
Polygon layers / 3D texture
Polygon faces / vertex colors
Polygon faces / 3D texture

Conclusions

All scene data must be built and stored on the host, then transferred in pieces to the GPU for rendering. The maximum scene size is limited by the amount of host physical memory, not by the GPU. Different types of meshes and different rendering styles for those meshes has a substantial impact on the amount of memory required to create and fill vertex buffers to draw the meshes.

Name	Changed	Size
Data	Fri, 09/09/2016 - 14:09	--
Images	Fri, 09/09/2016 - 14:05	--

Path hierarchy breadcrumbs

Folder description with rich text and metadata

File/data browser

Project information

Website: https://dibbs.seedme.org

Why build a platform?

Research computing is highly collaborative, distributed and often uses disparate compute resources. Currently available tools do not meet sharing and collaborative needs that must collocate **d**ata, **d**escription and **d**iscussion (3D) and additionally handle transfer, storage and access control. Furthermore, these tools must be cross platform and readily pluggable for automation with existing scientific workflows.

For whom?

Computing researchers
- Collaboration hub
- Personal dashboard

Developers
Integrate with scientific applications

Project repositories

Gateways
Service for data sharing, data publishing, data escrow

CI providers: Offer the platform to your user base

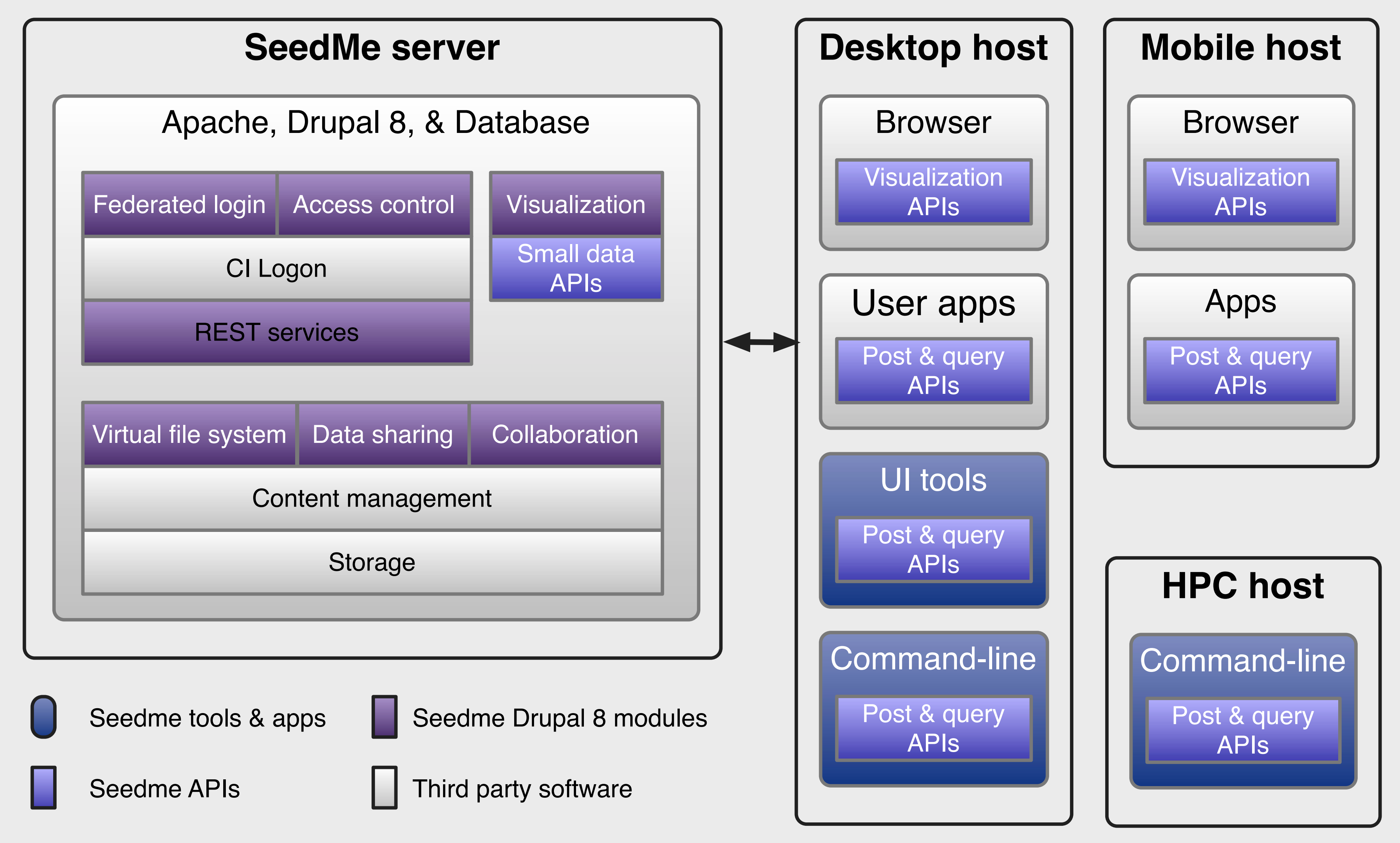
Use how?

As a cloud service
- dibbs.seedme.org
- www.seedme.org

DIY - Run own instance
- On your own hardware
- Condo hardware

Provider run instances
- At your institution
- At national centers
- At public cloud

~~No lock in~~



SeedMe server
Apache, Drupal 8, & Database
Federated login
Access control
Visualization
CI Logon
Small data APIs
REST services
Virtual file system
Data sharing
Collaboration
Content management
Storage

Desktop host
Browser
Visualization APIs
User apps
Post & query APIs
UI tools
Post & query APIs
Command-line
Post & query APIs

Mobile host
Browser
Visualization APIs
Apps
Post & query APIs

HPC host
Command-line
Post & query APIs

Seedme tools & apps
Seedme Drupal 8 modules
Seedme APIs
Third party software

- Cross-platform tools, APIs, and Drupal modules
- Post & query data from HPC jobs, workflows, apps, browsers, and command line
- Cloud file system for secure data sharing and collaboration
- Integrated lightweight visualization tools for quick analysis
- Secure access, sharing, and access controls

Acknowledgements: This work is supported by the National Science Foundation under Grant No. ACI-1443083. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF."