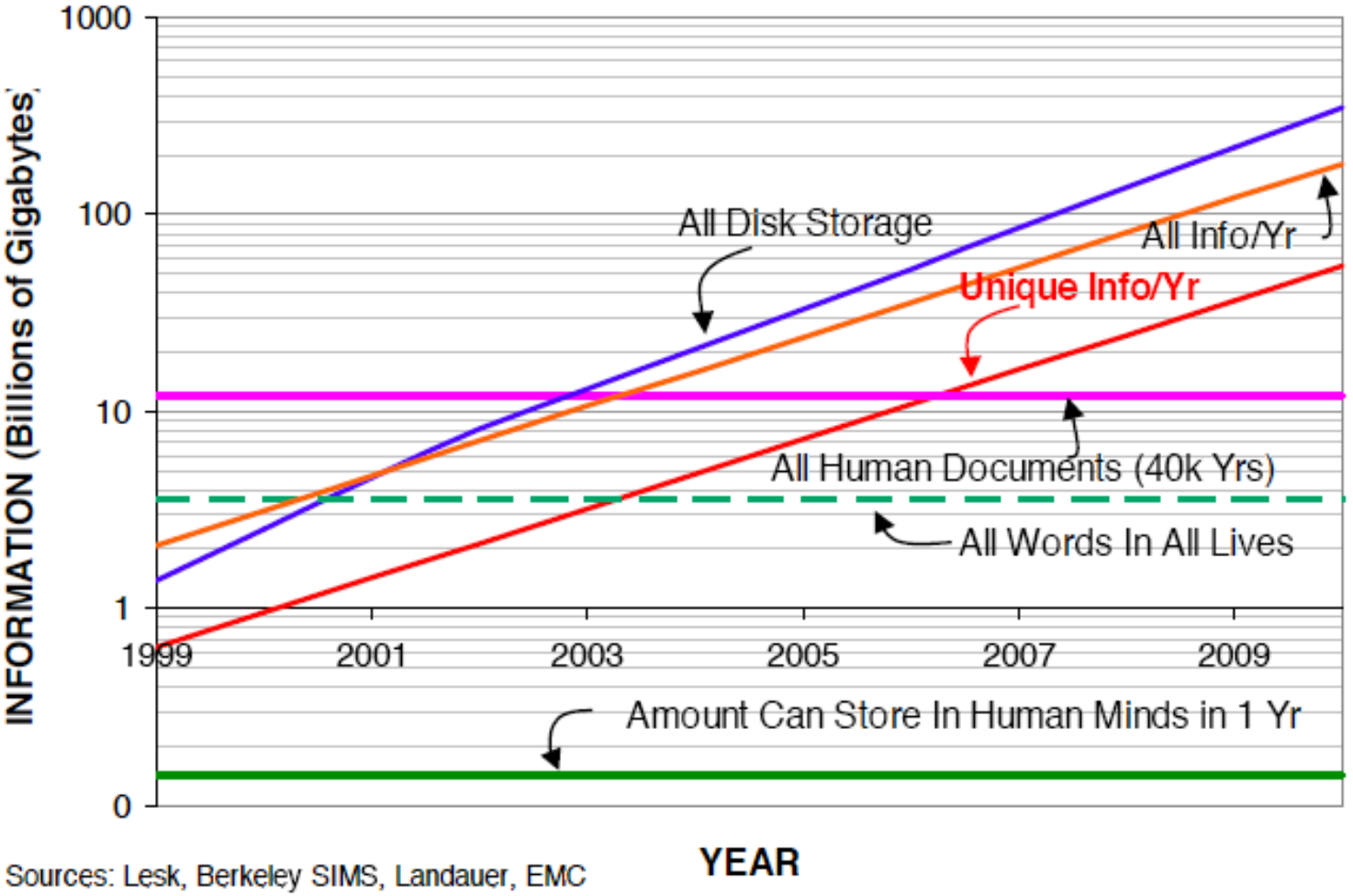
High-Performance Visualization

Thomas Fogal

Growth of data



Sources: Lesk, Berkeley SIMS, Landauer, EMC

VisIt / IceT

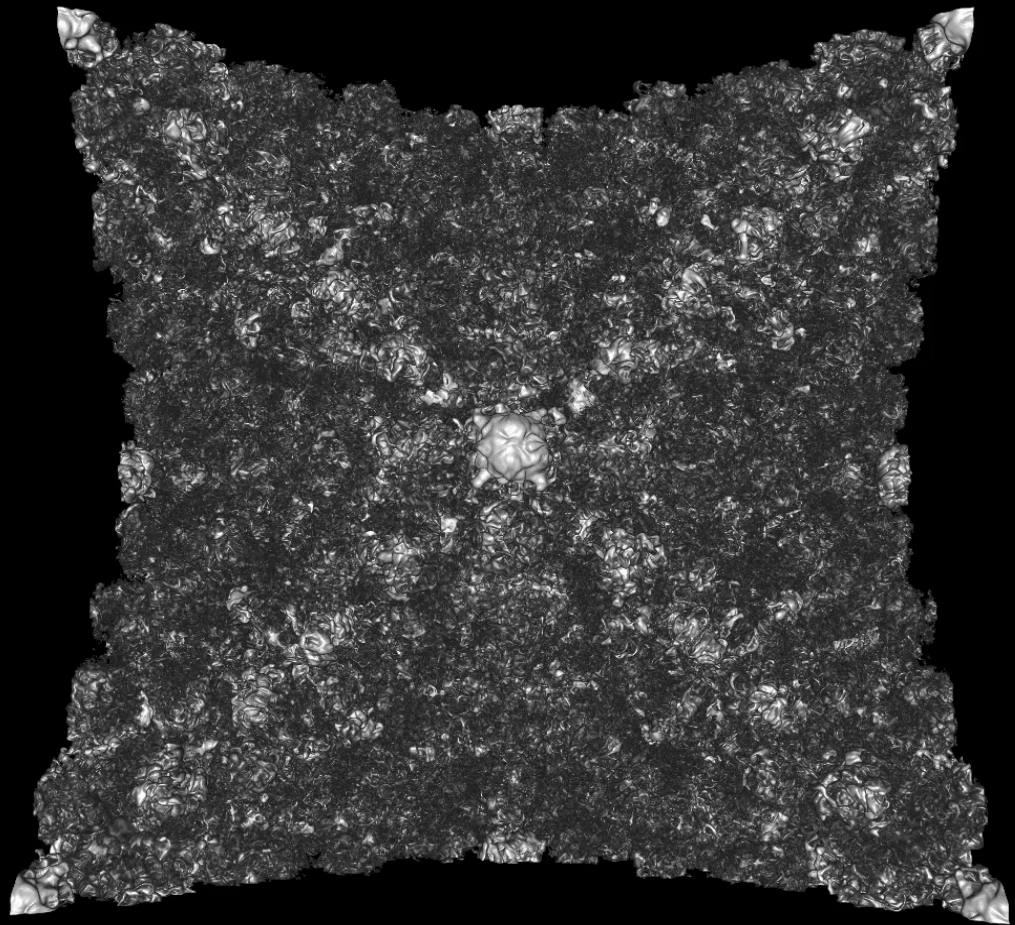
1 of 8 168.09%

VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data

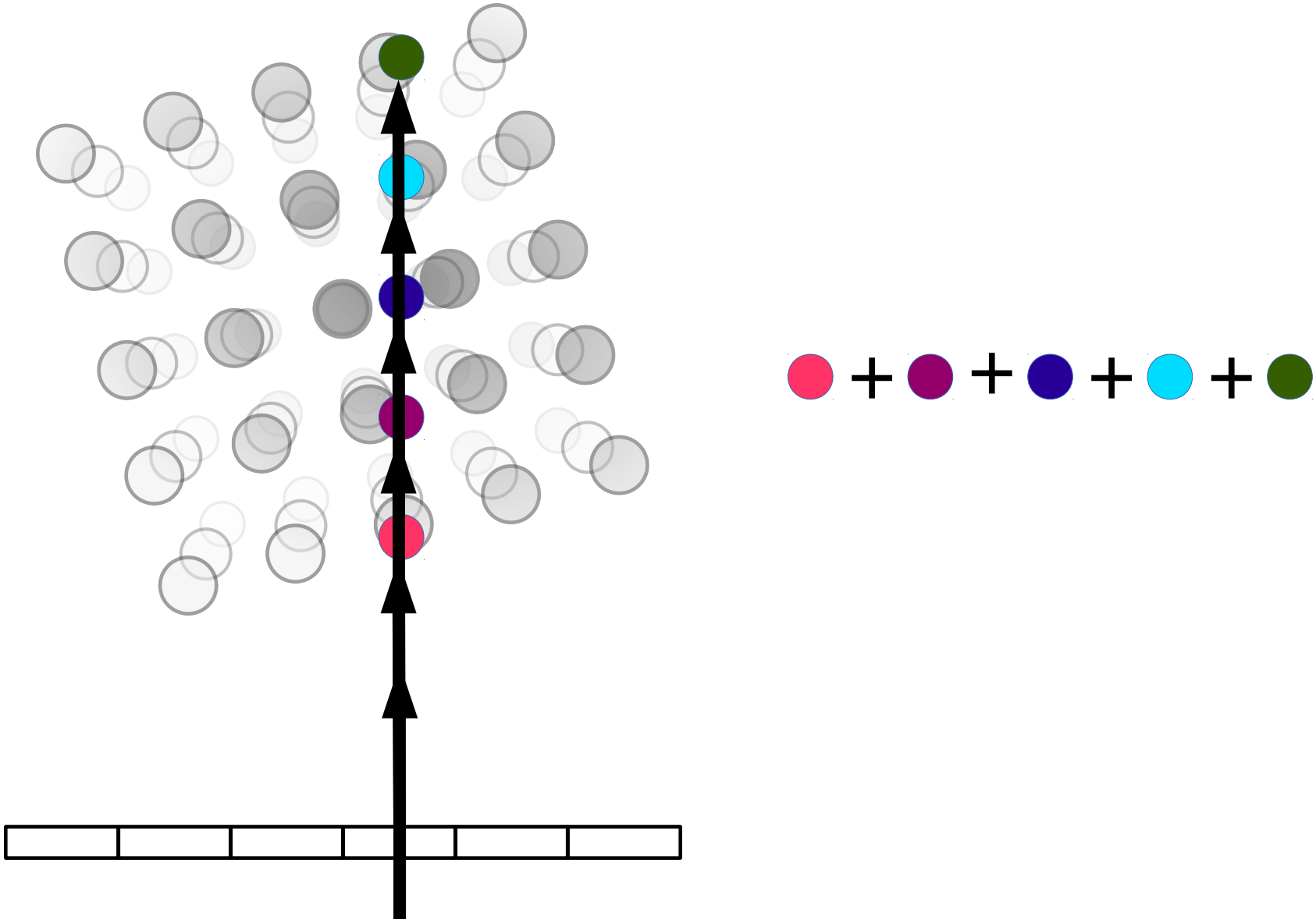
Hank Childs[*,§], Eric Brugger[†], Brad Whitlock[†], Jeremy Meredith[‡], Sean Ahern[‡], Kathleen Bonnell[†], Mark Miller[†], Gunther H. Weber[*], Cyrus Harrison[†], David Pugmire[‡], Thomas Fogal[¶], Christoph Garth[§], Allen Sanderson[¶], E. Wes Bethel[*], Marc Durant[▽], David Camp[*,§], Jean M. Favre[‖], Oliver Rübel[*], Paul Navrátil[△], Matthew Wheeler[α], Paul Selby[α], and Fabien Vivodtzev[±]

[*] Lawrence Berkeley National Laboratory, [†]Lawrence Livermore National Laboratory, [‡]Oak Ridge National Laboratory, [§]University of California at Davis, [¶]University of Utah, [▽]Tech-X Corporation, [‖]Swiss National Supercomputing Center, [△]Texas Advanced Computing Center, [α]Atomic Weapons Establishment, [±]French Atomic Energy Commission, CEA/CESTA

Abstract. VisIt is a popular open source tool for visualizing and analyzing data. It owes its success to its foci of increasing data understanding, large data support, and providing a robust and usable product, as well as its underlying design that fits today's supercomputing landscape. In this short paper, we describe the VisIt project and its accomplishments.

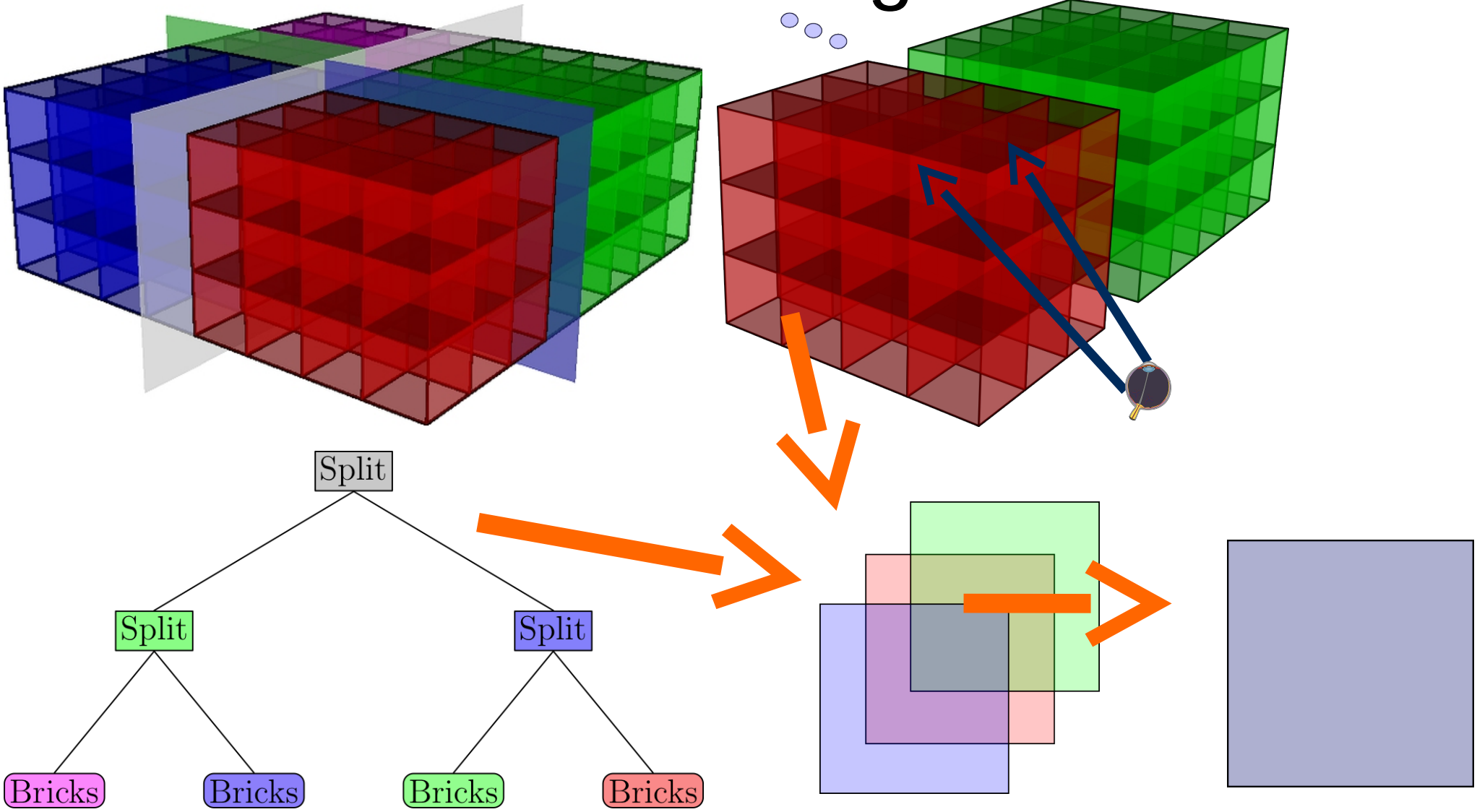


VolRen Background



Thanks: Florian Hoffman

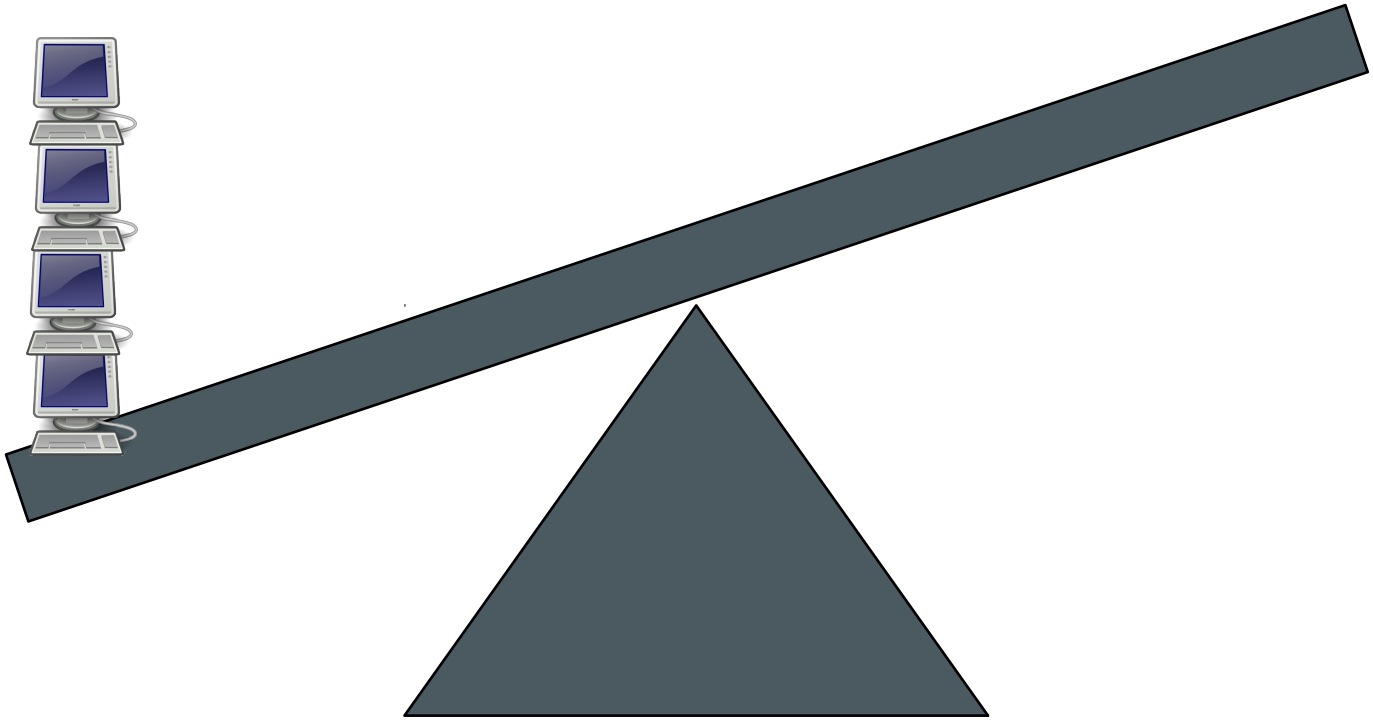
Traditional large-scale volume rendering

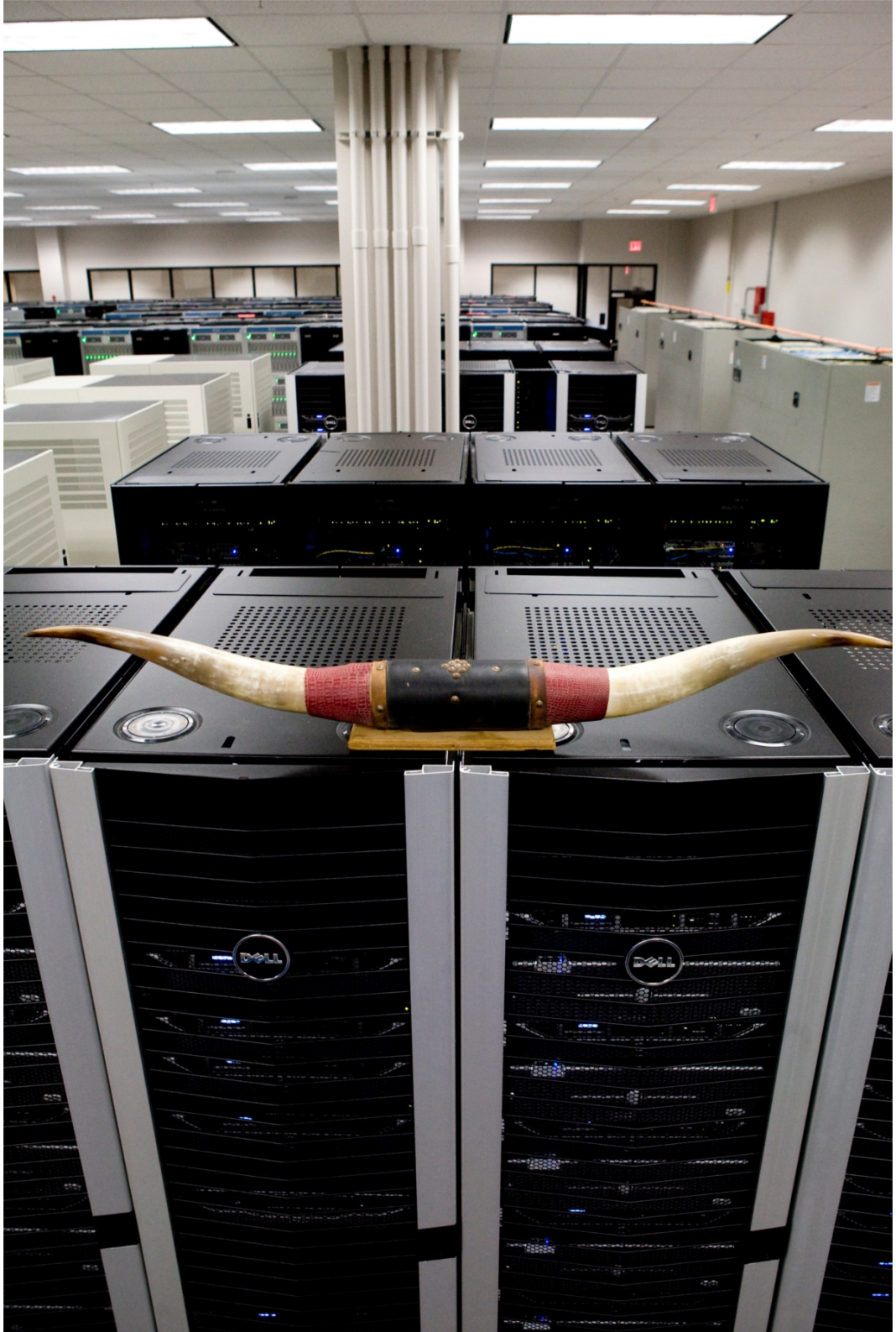


Compositing scalability

Rendering

Compositing

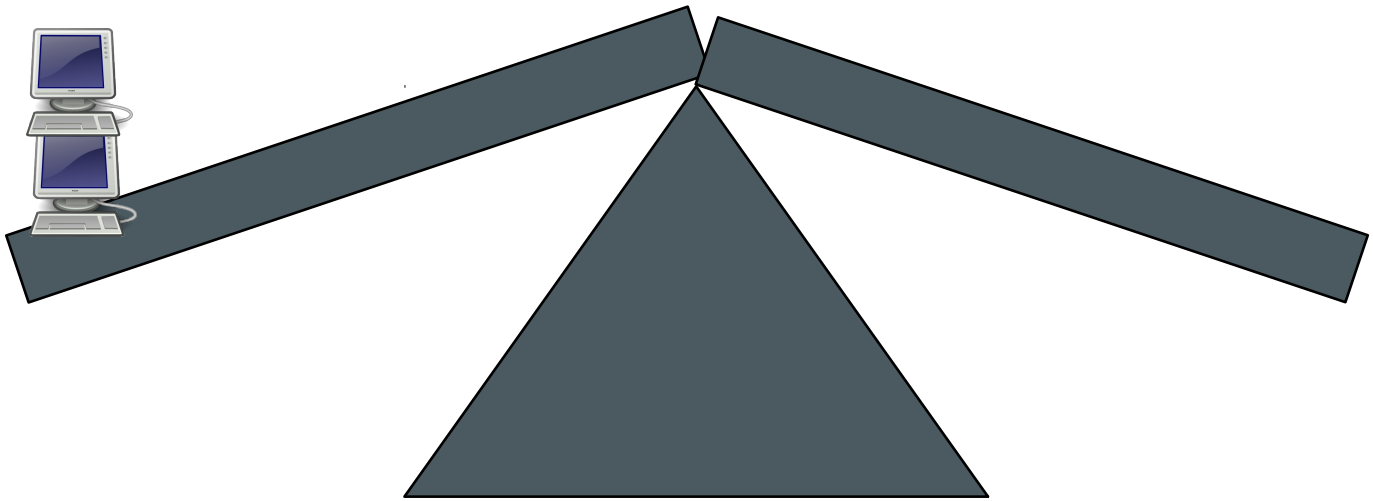



DELL
DELL

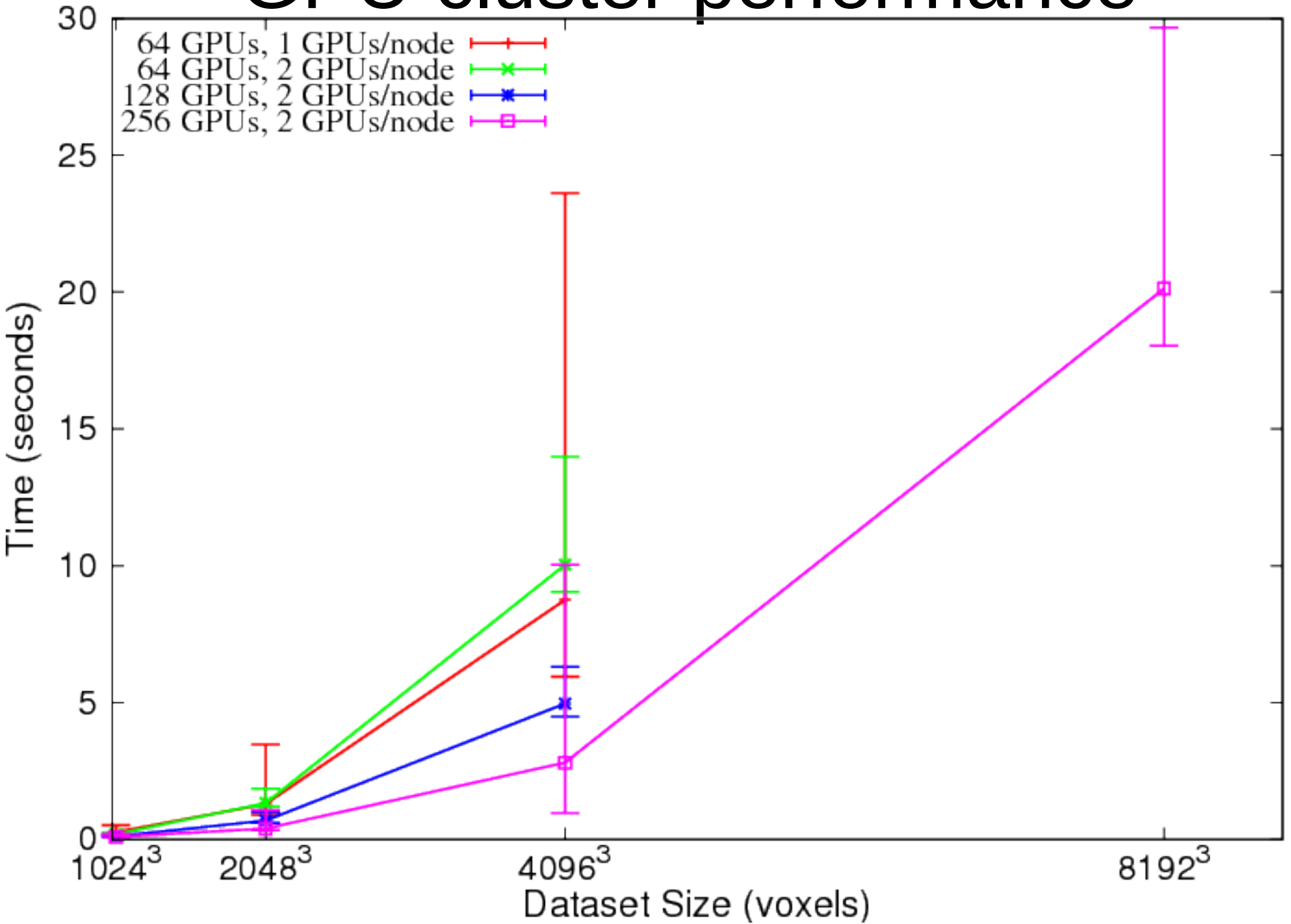
Compositing scalability

Rendering

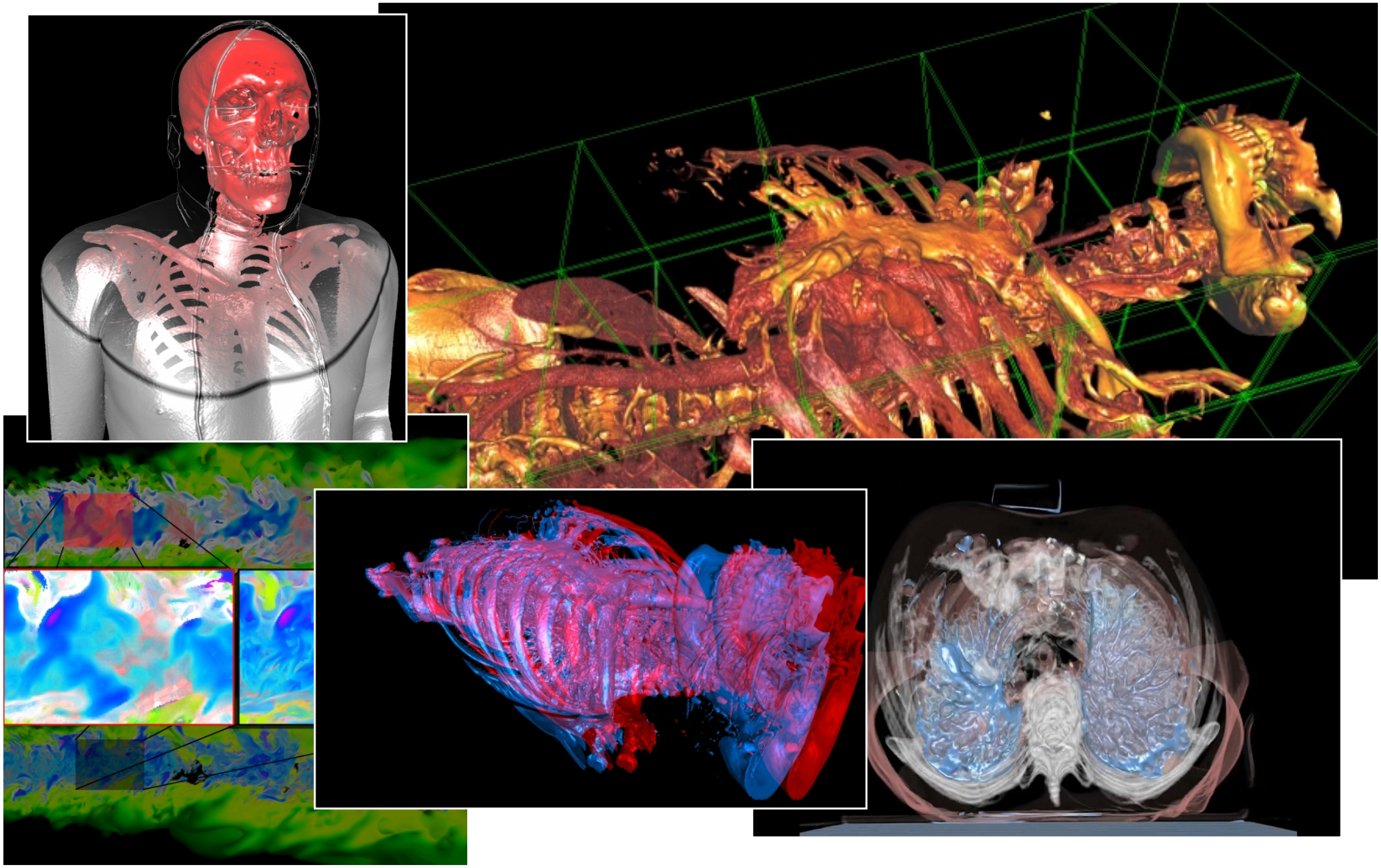
Compositing



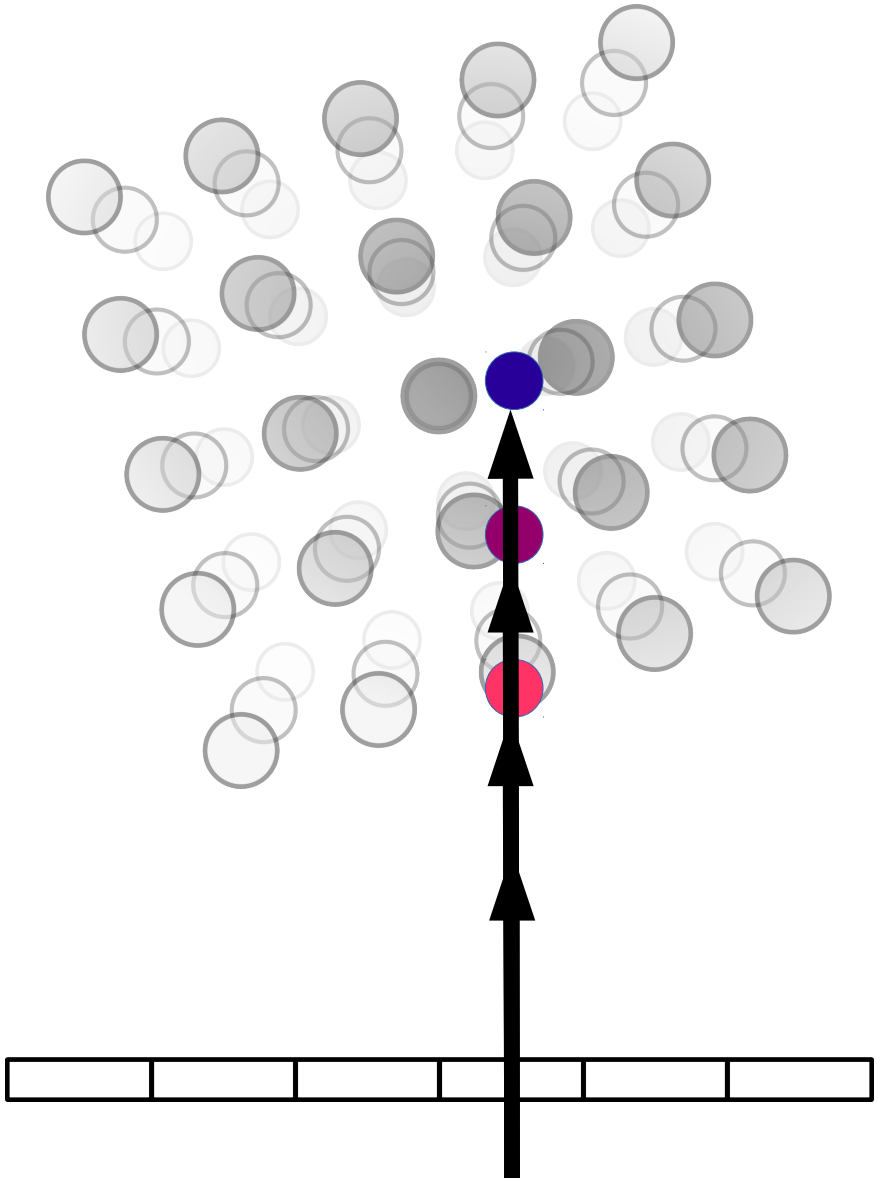
GPU cluster performance



Single node volume visualization

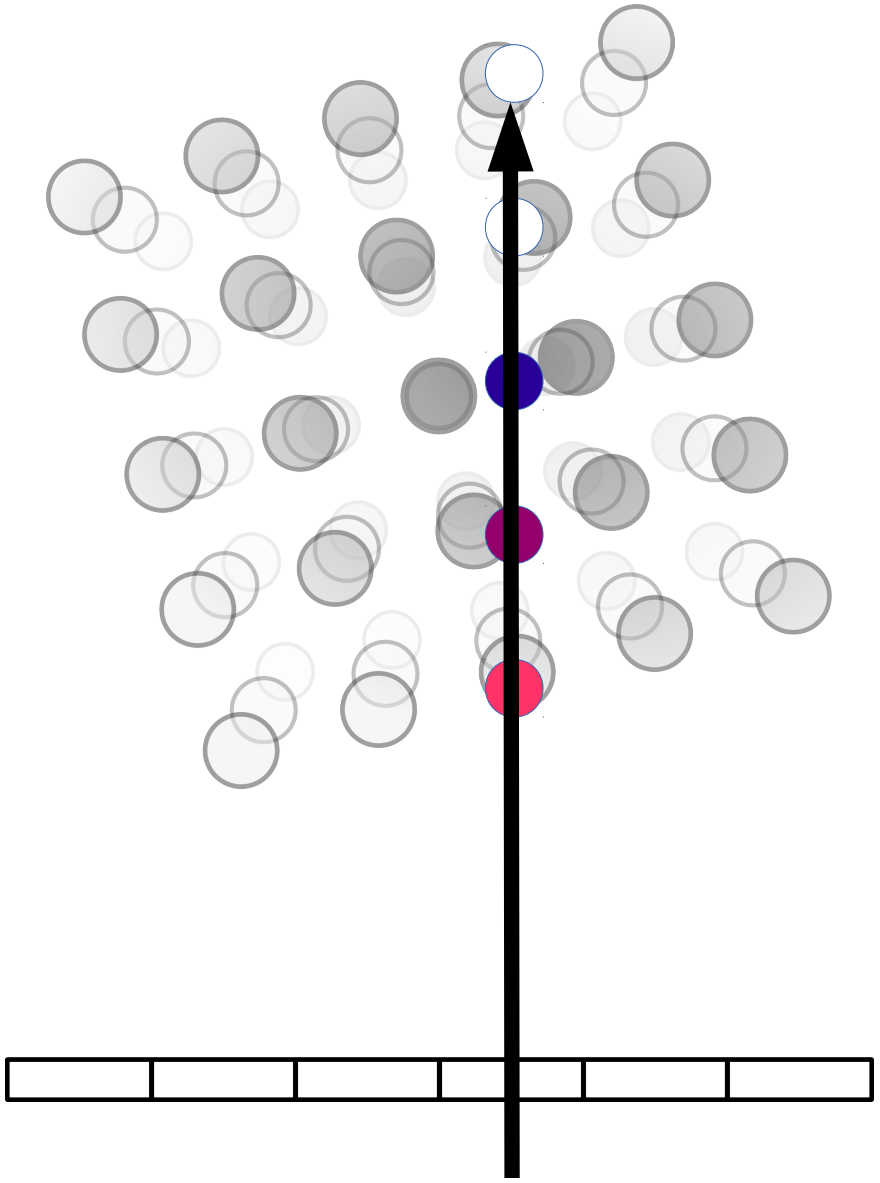


Ray saturation



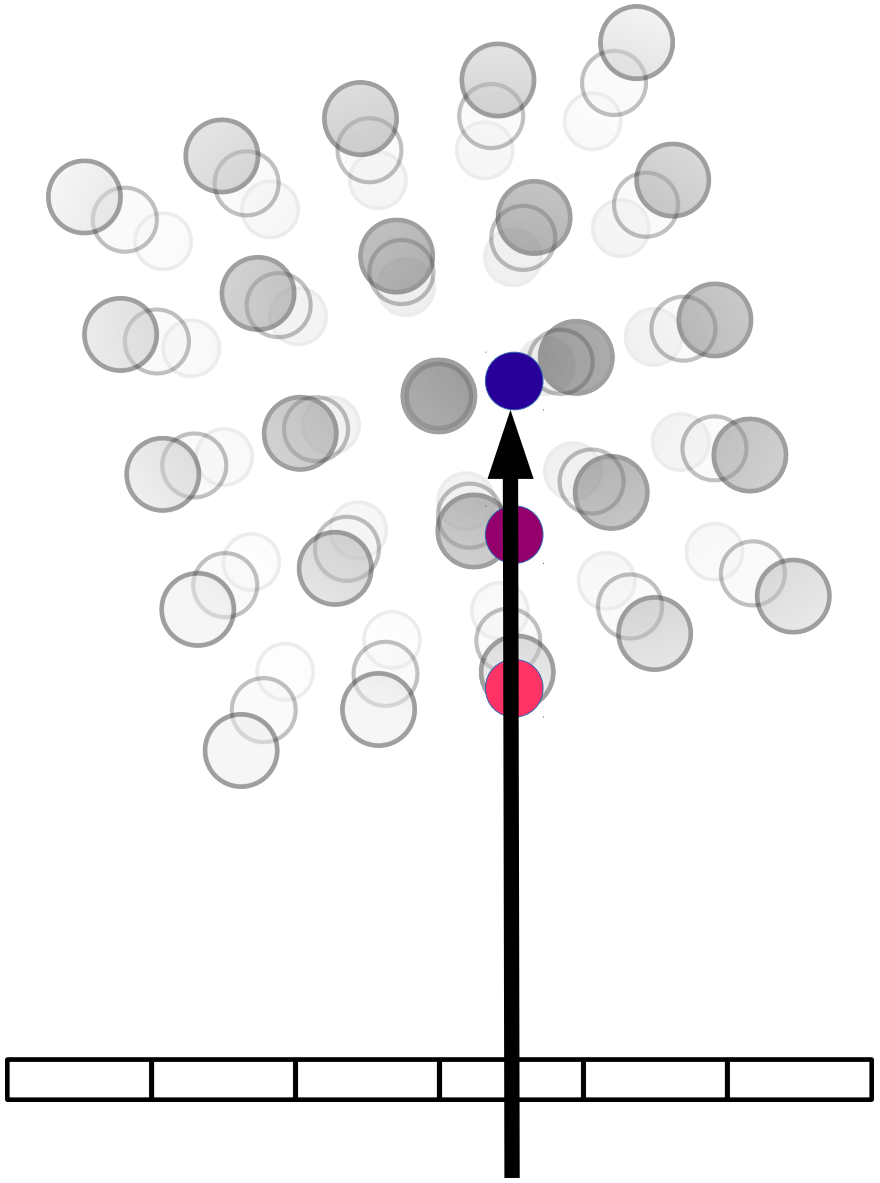
Thanks: Florian Hoffman

Full Alpha



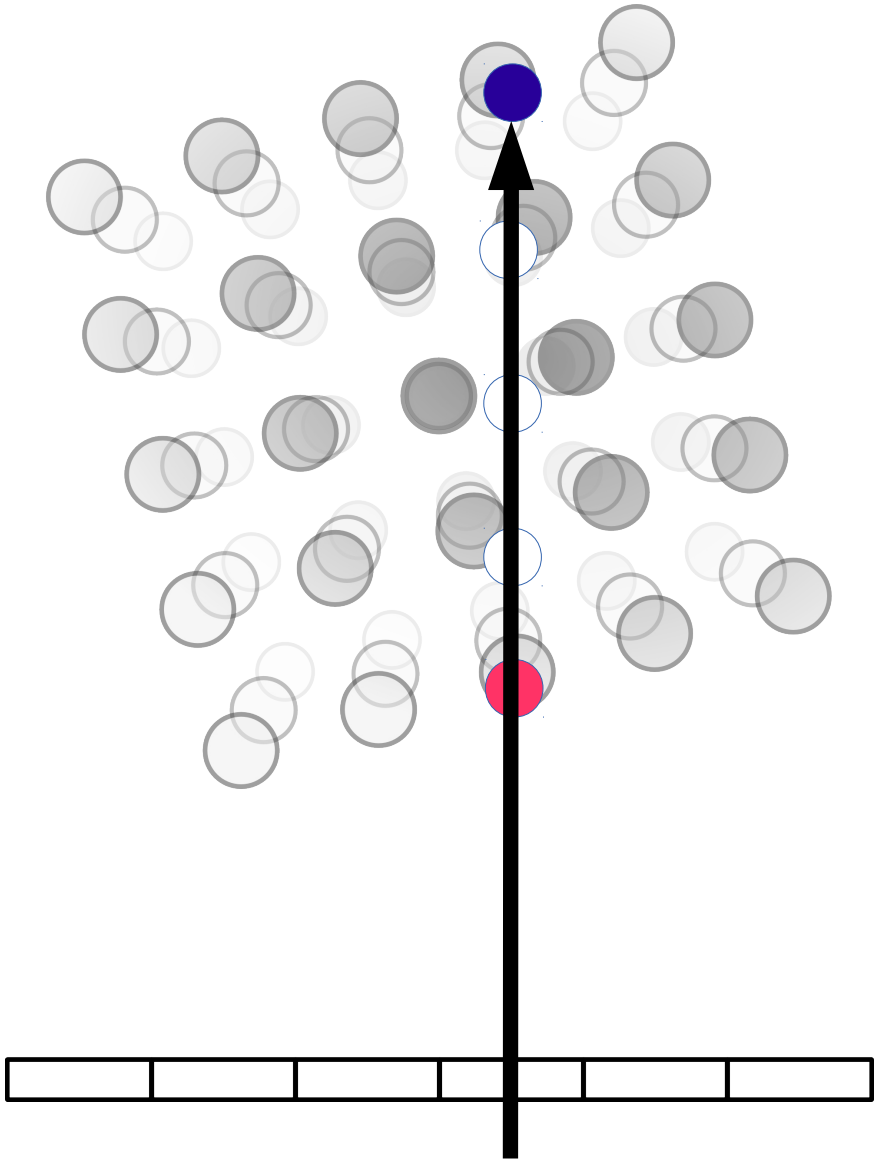
Thanks: Florian Hoffman

Early Ray Termination



Thanks: Florian Hoffman

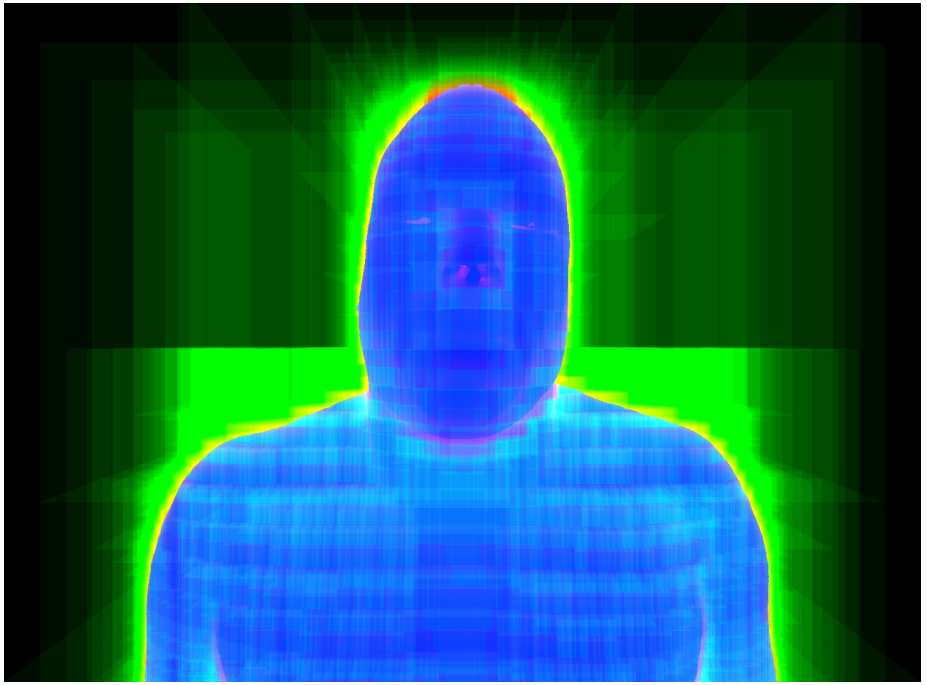
Empty Space Leaping



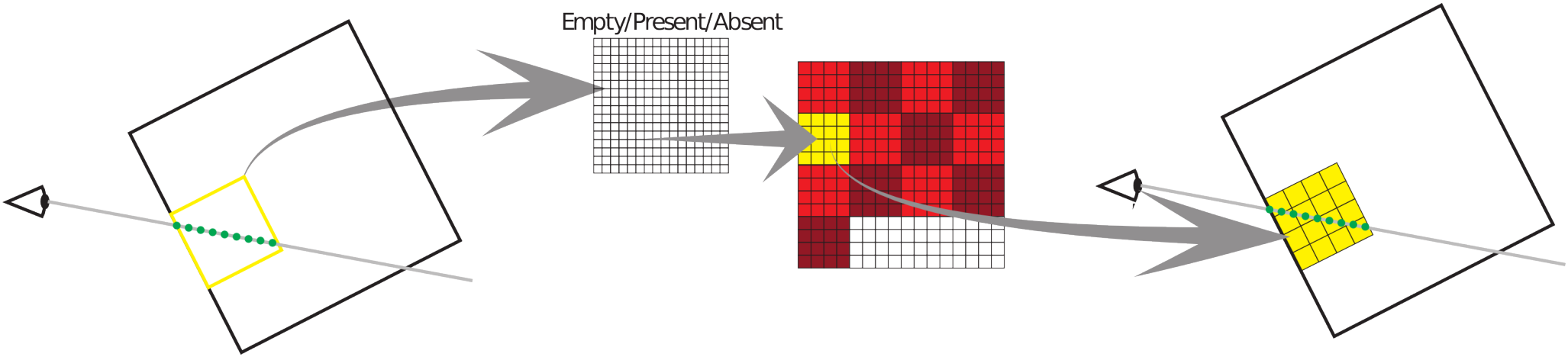
Thanks: Florian Hoffman

What's important for performance

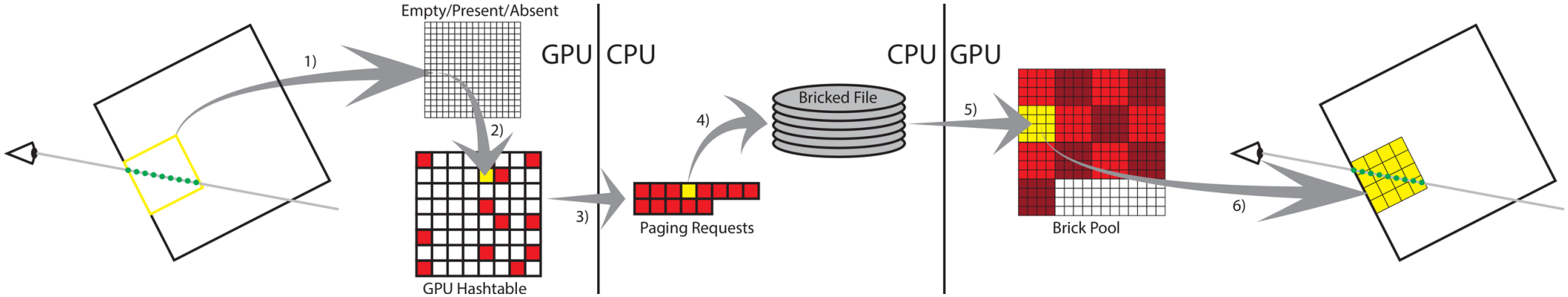
- Identifying densely-sampled regions
- Transition to coarse sampling quickly
- Communicate data needed to IO



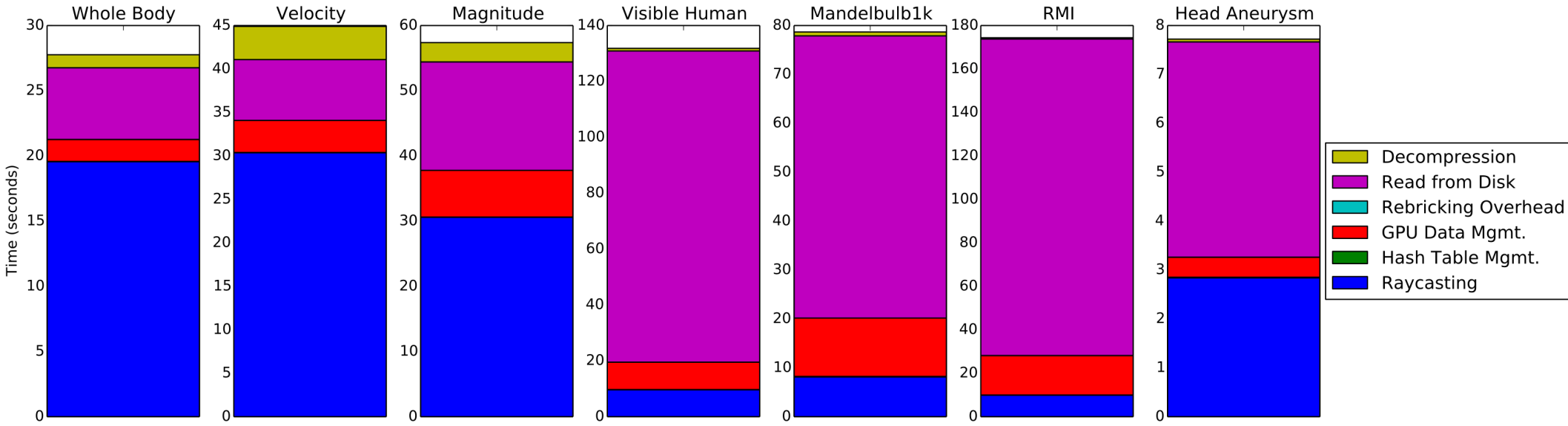
Ray-Guided Rendering



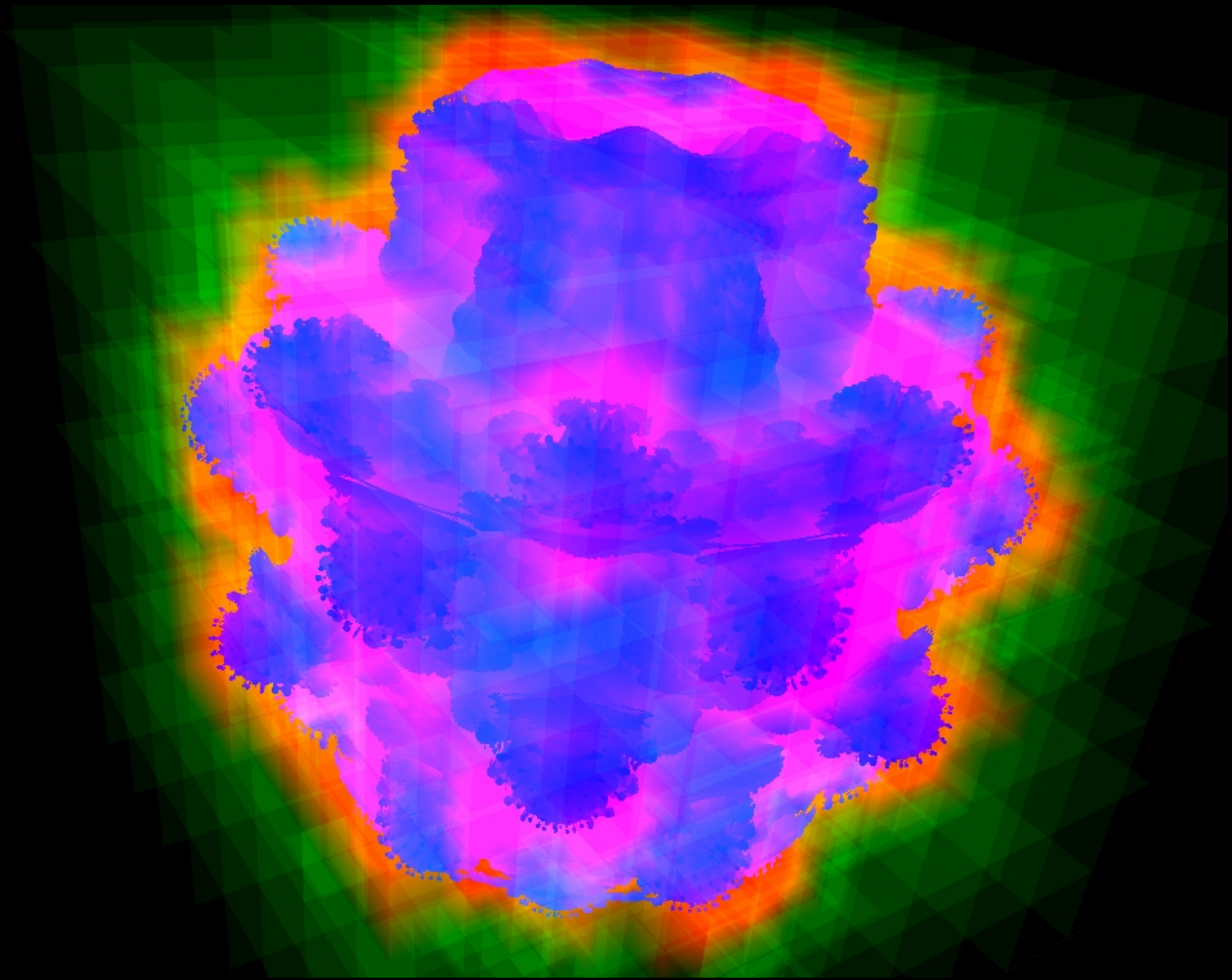
Ray-Guided Rendering



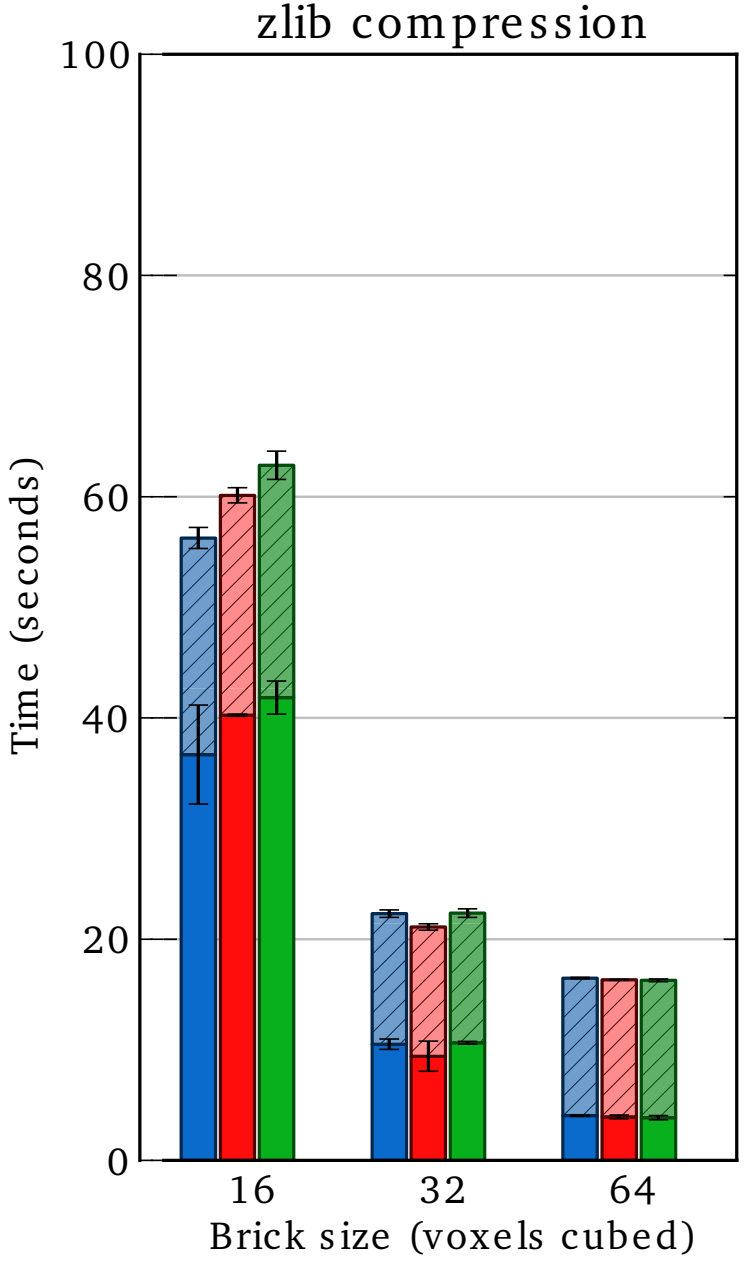
Where does the time go?



Brick Size



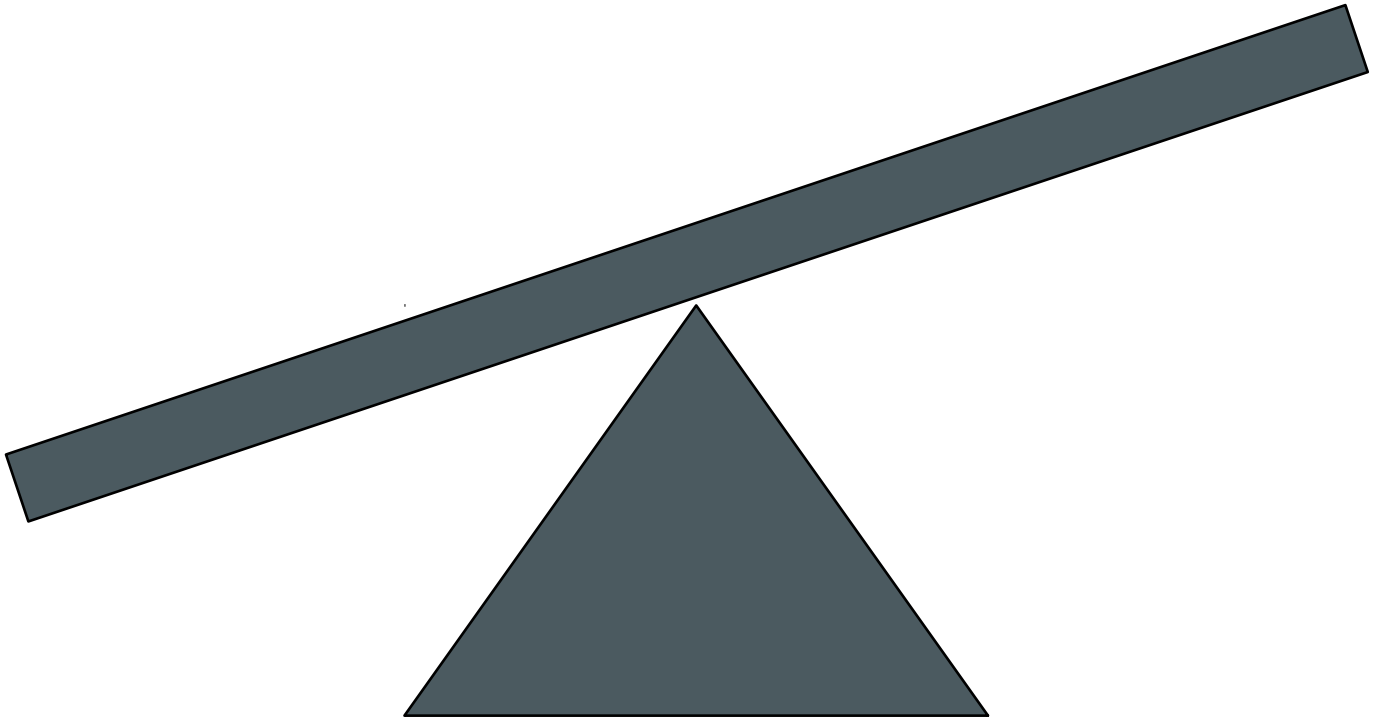
Brick Size: IO



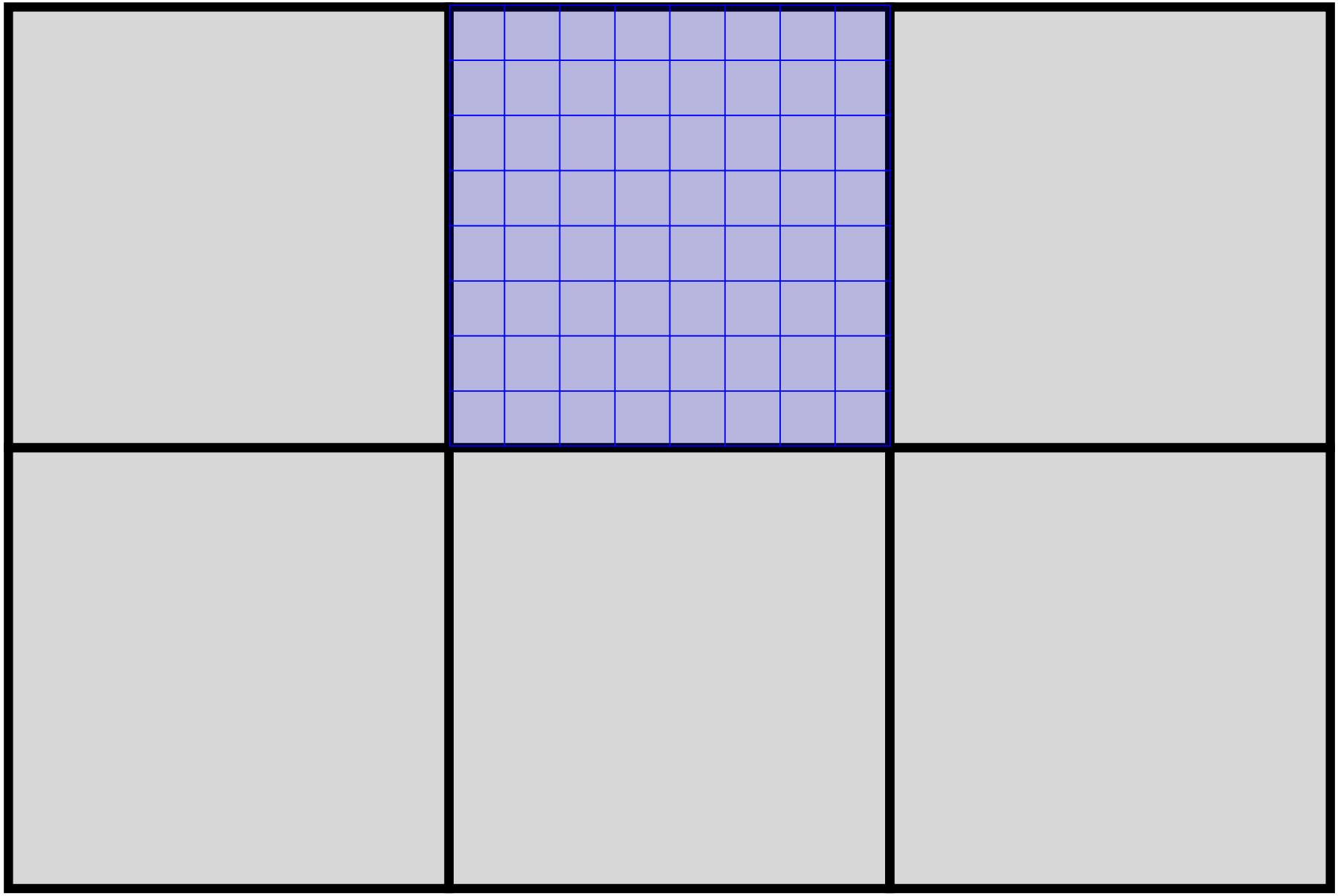
Brick size balancing

Render performance

IO performance



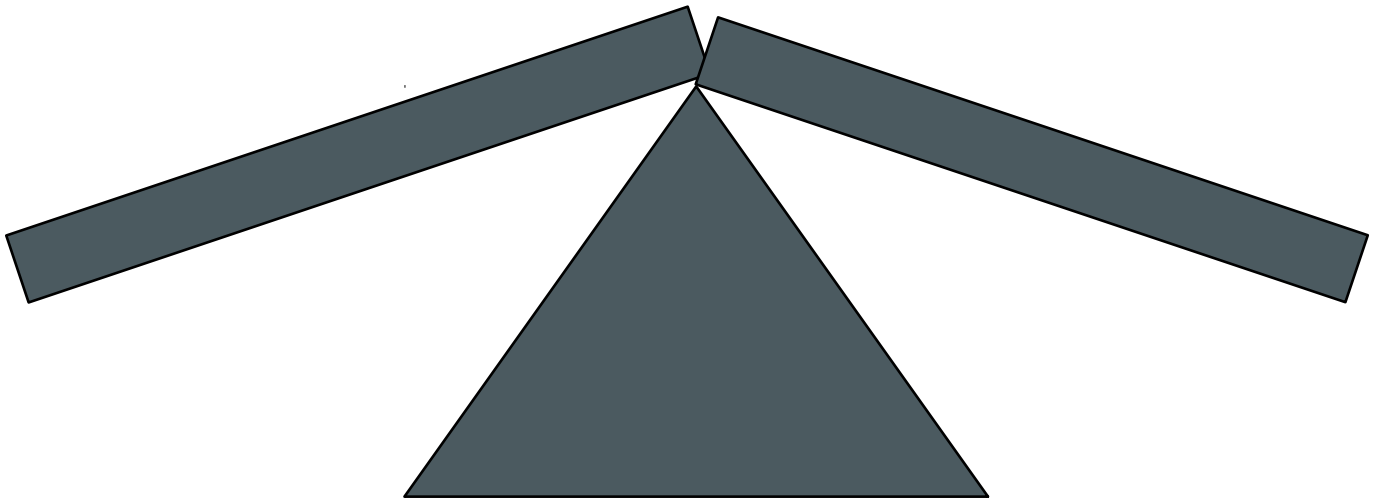
Dynamic Bricking



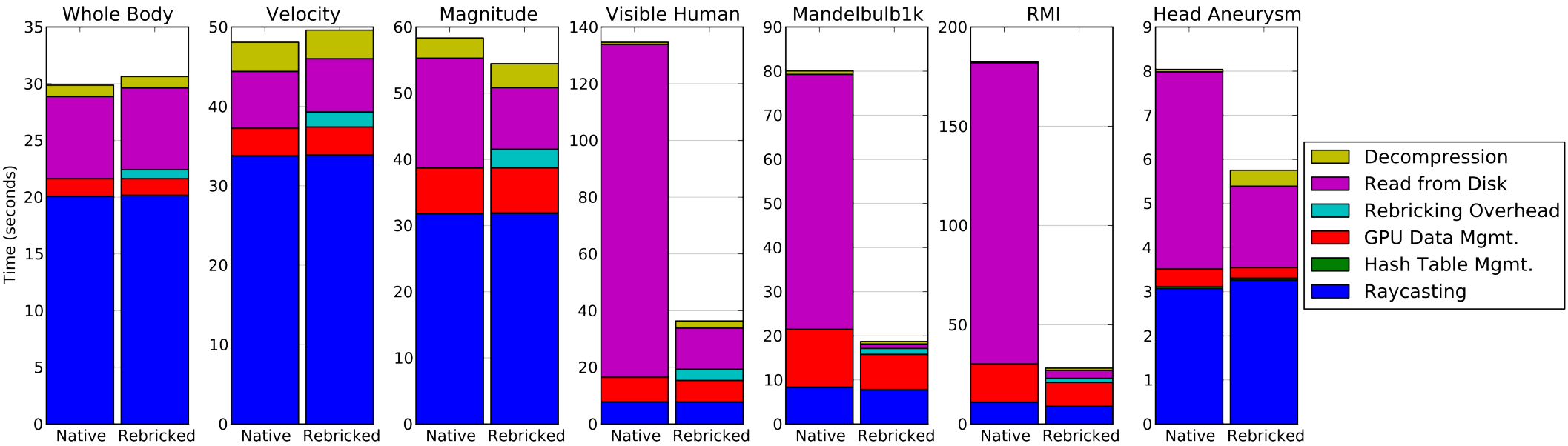
Dueling subsystems

Rendering

IO

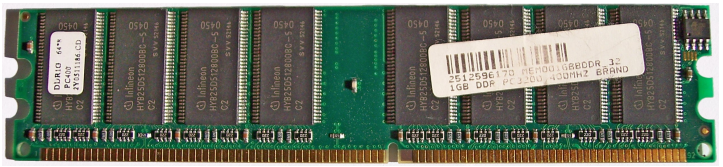


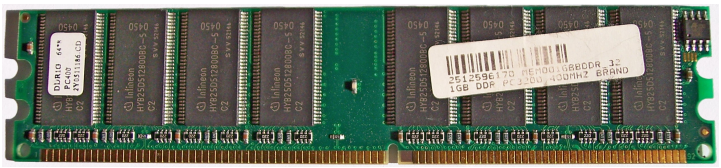
Dynamic bricking reduces I/O



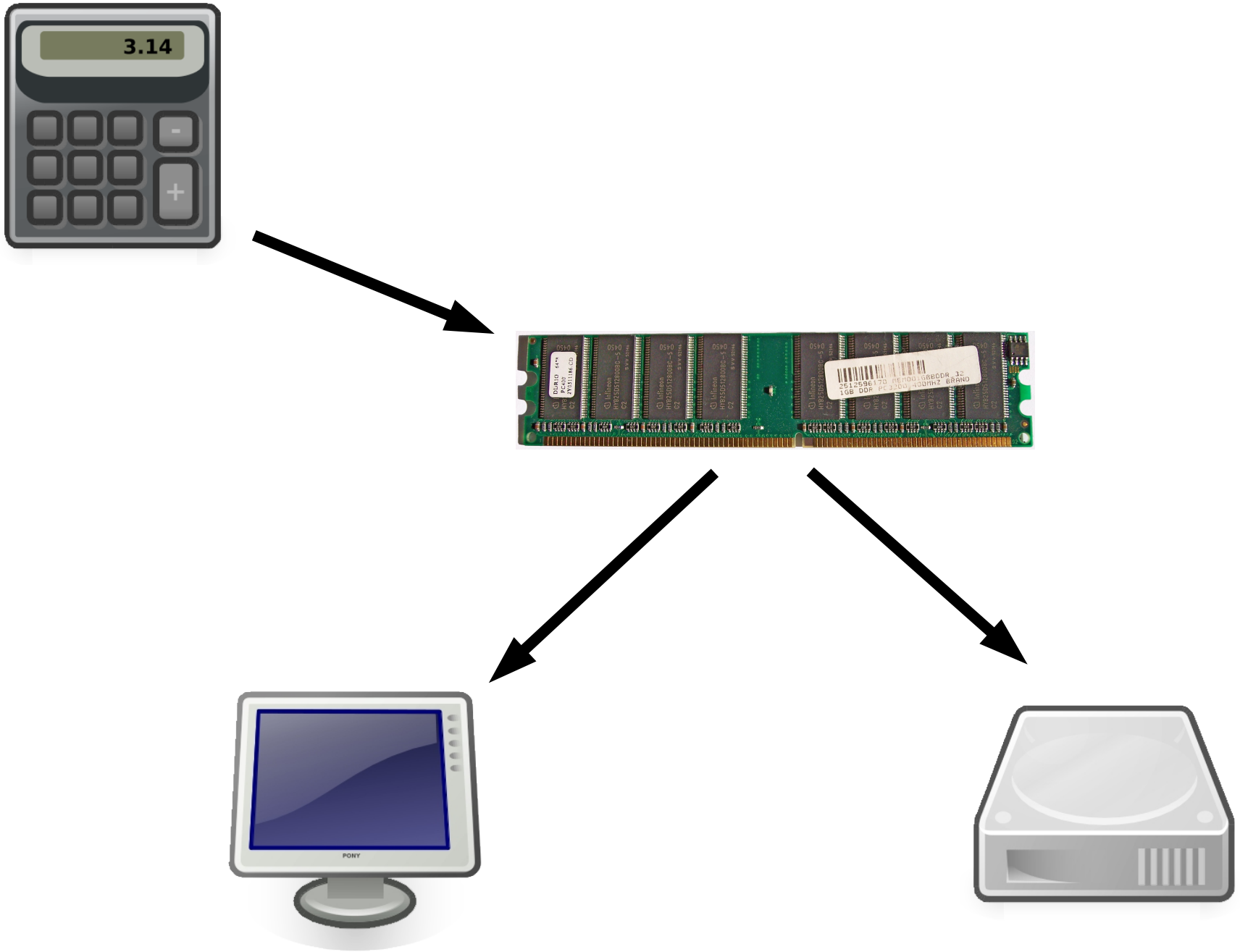
In situ visualization


3.14






PONY

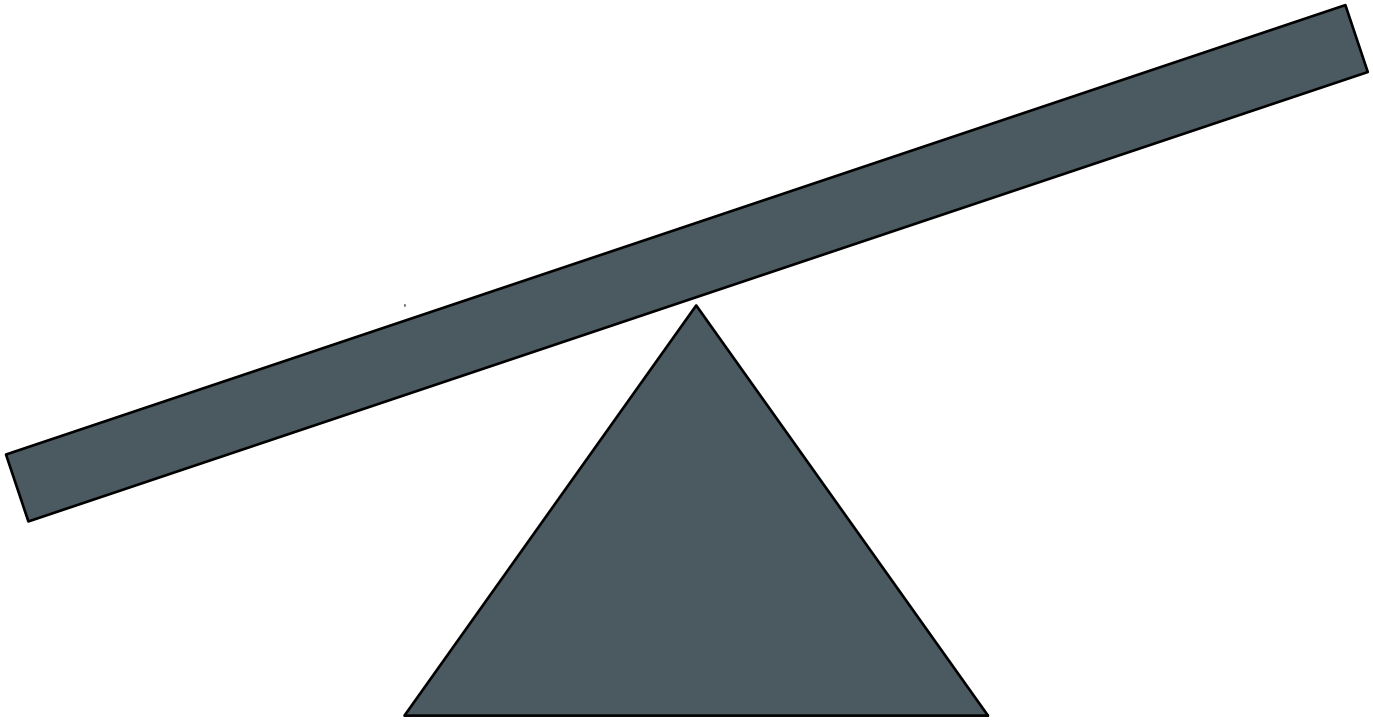

3.14
PONY

Story time

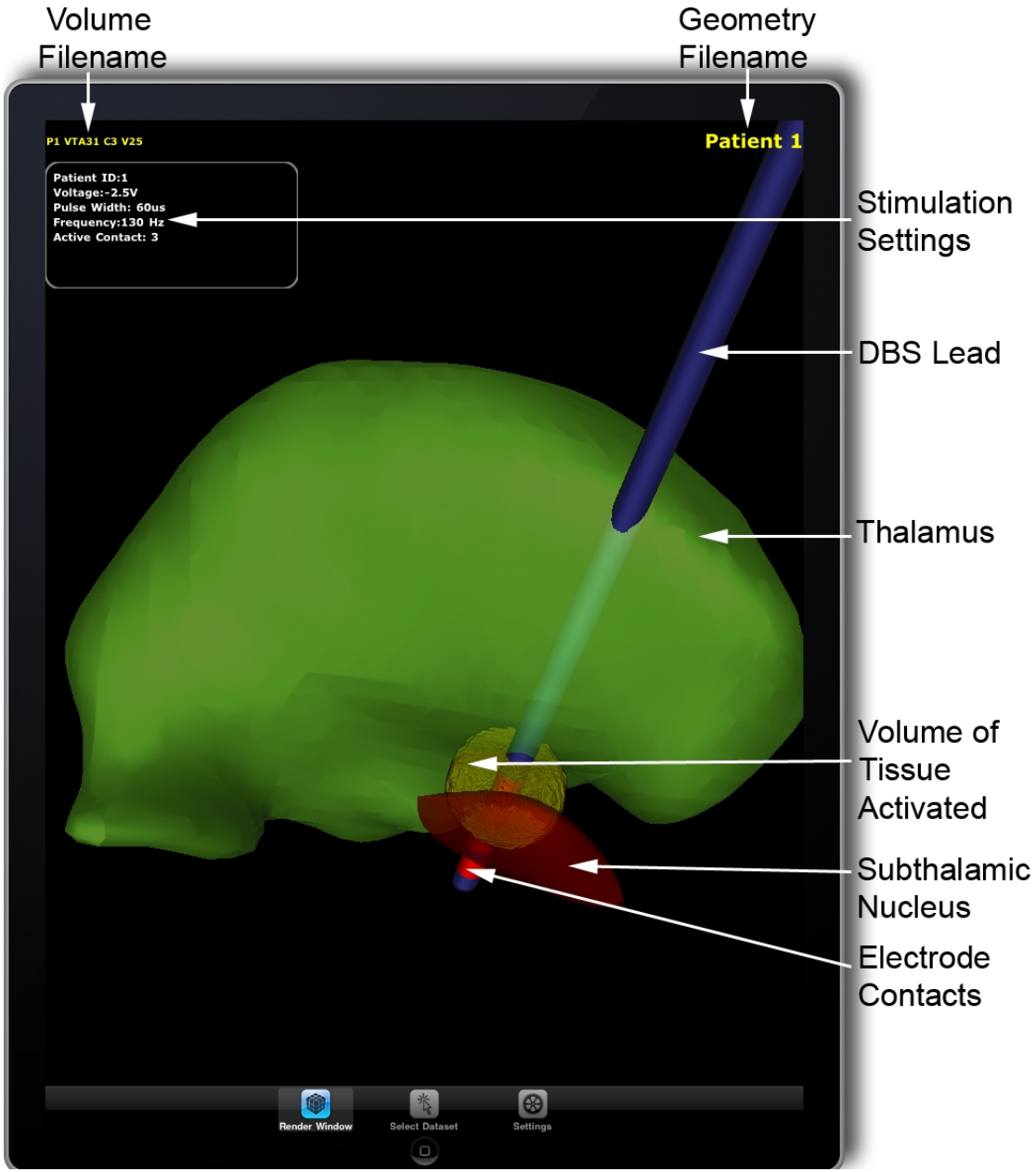
Tradeoffs

Working in situ vis

Seeing the light of day



Simplicity


Volume Filename
Geometry Filename
P1 VTA31 C3 V25
Patient 1
Patient ID:1
Voltage:-2.5V
Pulse Width: 60us
Frequency:130 Hz
Active Contact: 3
Render Window
Select Dataset
Settings
Stimulation Settings
DBS Lead
Thalamus
Volume of Tissue Activated
Subthalamic Nucleus
Electrode Contacts

Where to insert visualization

```
gdb -q ./a.out \
-ex "break fopen" \
-ex "break main" \
-ex "run" \
-ex "condition 1 strstr(filename, \".csv\")" \
-ex "disable 2" \
-ex "cont" \
-ex "up"
```

Where to insert visualization

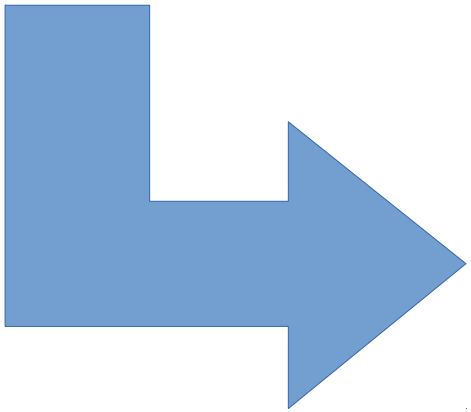
```
gdb -q ./a.out \
-ex "break fopen" \
-ex "break main" \
-ex "run" \
-ex "condition 1 strstr(filename, \".csv\")" \
-ex "disable 2" \
-ex "cont" \
-ex "up"
```

Where to insert visualization

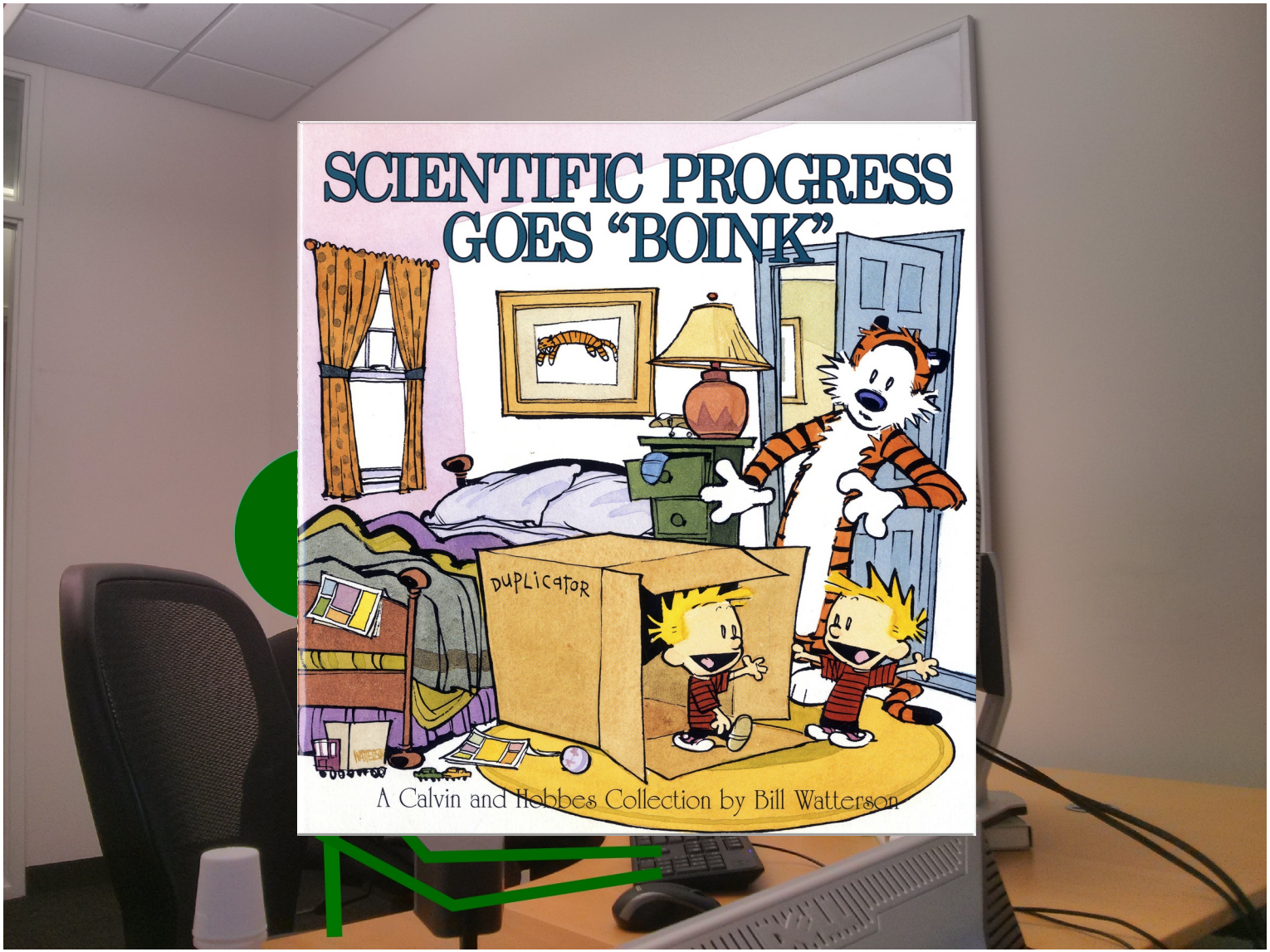
```
gdb -q ./a.out \
-ex "break fopen" \
-ex "break main" \
-ex "run" \
-ex "condition 1 strstr(filename, \".csv\")" \
-ex "disable 2" \
-ex "cont" \
-ex "up"
```

Where to insert visualization

```
gdb -q ./a.out \
-ex "break fopen" \
-ex "break main" \
-ex "run" \
-ex "condition 1 strstr(filename, \".csv\")" \
-ex "disable 2" \
-ex "cont" \
-ex "up"
```

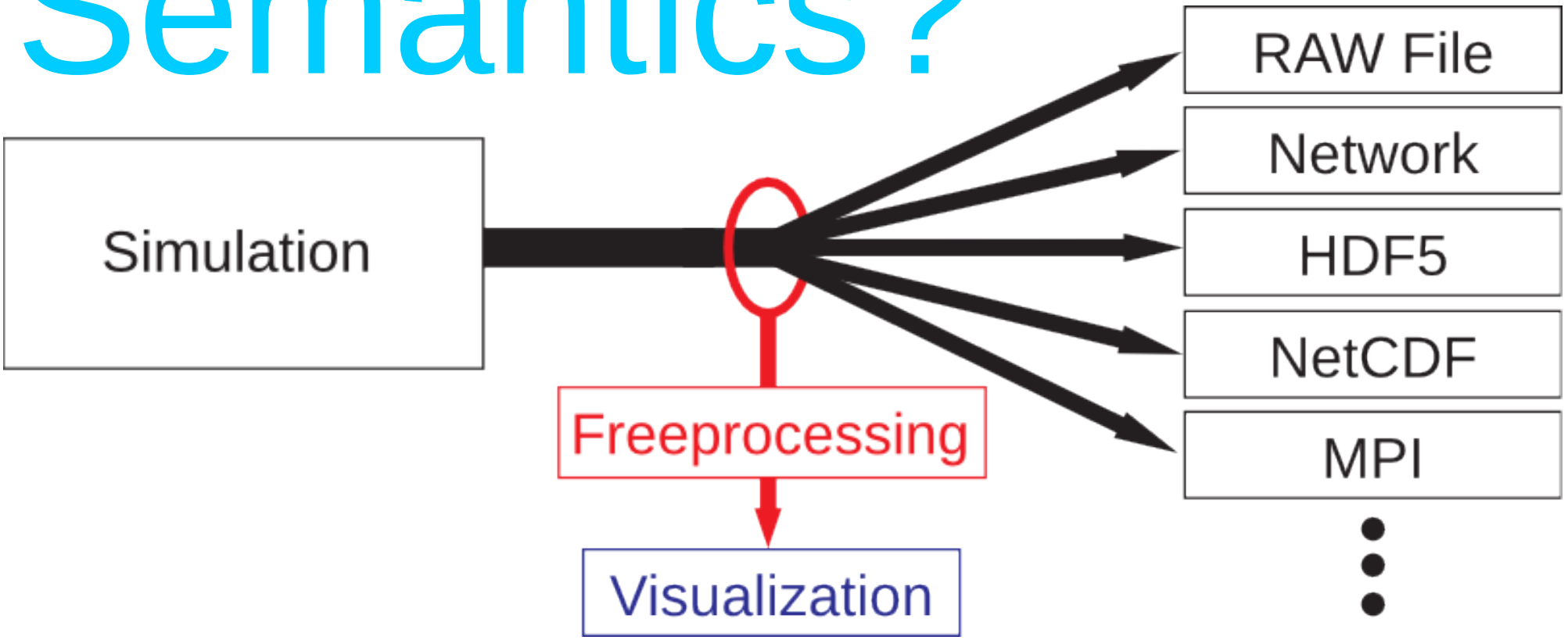


Filename.c:1204


SCIENTIFIC PROGRESS GOES "BOINK"
DUPLICATOR
A Calvin and Hobbes Collection by Bill Watterson

Interception

Semantics?

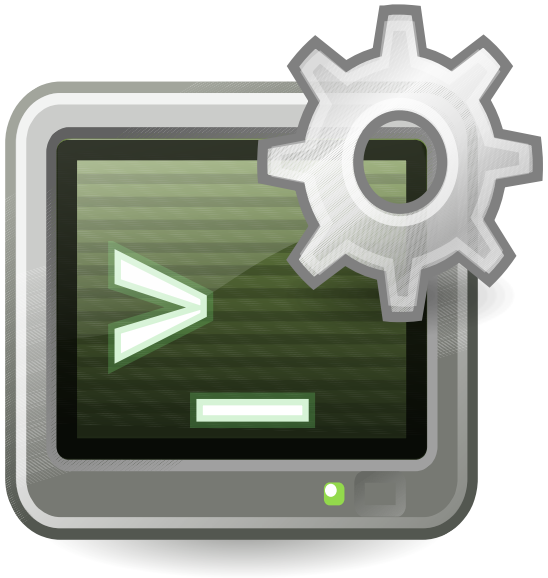


"Inside every large problem is a small problem struggling to get out."

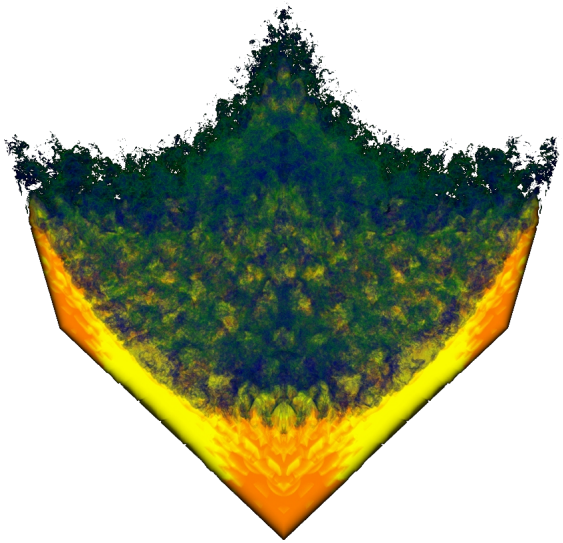
– Tony Hoare

Data *model* + Program

{
  3D,
  isotropic
}



=



ImageVis3D

```
for (uint32_t z=iOverlap; z < bricksize.z-iOverlap; z++)
for (uint32_t y=iOverlap; y < bricksize.y-iOverlap; y++)
for (uint32_t x=iOverlap; x < bricksize.x-iOverlap; x++)
{
  DOUBLEVECTOR3 vGradient = ComputeGradient(
    pTempBrickData, normalizationFactor, iCompcount,
    bricksize, x,y,z);
  if(vGradient.length() > fMaxGradMagnitude)
    fMaxGradMagnitude = vGradient.length();
}
```

Histogram2DDataBlock.cpp

LAMMPS

```
for(i=0; i<subNbx+3; i++)
  for(j=0; j<subNby+3; j++)
    for(k=0; k<subNbz+3; k++) {
      u_lb[i][j][k][0]=0.0;
      u_lb[i][j][k][1]=0.0;
      u_lb[i][j][k][2]=0.0;
      density_lb[i][j][k] = densityinit;
    }
```

fix_lb_fluid.cpp

libgfortran

```
for (y = 0; y < ycount; y++) {
  bbase_y = &bbase[y*bystride];
  s = (GFC_INTEGER_16) 0;
  for (n = 0; n < count; n++)
    s += abase[n*axstride] *
         bbase_y[n*bxstride];
  dest[y*rxstride] = s;
}
```

matmul_i4.c

Enzo

```
for (int k = k1; k <= k2; k++) {
  for (int j = j1; j <= j2; j++) {
    for (int i = i1; i <= i2+1; i++) {
      int idx3d = (k*jdim+j)*idim + i;
      float vdiff1 = 0.0f, wdiff1 = 0.0f;
      if (lj1)
          vdiff1 = (vslice[idx3d-dimx] +
vslice[idx3d-1-dimx])
              - (vslice[idx3d+dimx] +
vslice[idx3d-1+dimx]);
```

cuPPM.cuh

PsiPhi

```
DO k = 1, nK
  DO j = 1, nJ
    DO i = 1, nI
      velFluctDisp = velFluctDisp +
        (VelFluct(i,j,k) - meanDbl)**2
    END DO
  END DO
END DO
```

inflow.f90

Enzo

```
for (int k = k1; k <= k2; k++) {
  for (int j = j1; j <= j2; j++) {
    for (int i = i1; i <= i2+1; i++) {
      int idx3d = (k*jdim+j)*idim + i;
      float vdiff1 = 0.0f, wdiff1 = 0.0f;
      if (lj1)
          vdiff1 = (vslice[idx3d-dimx] +
vslice[idx3d-1-dimx])
              - (vslice[idx3d+dimx] +
vslice[idx3d-1+dimx]);
```

cuPPM.cuh

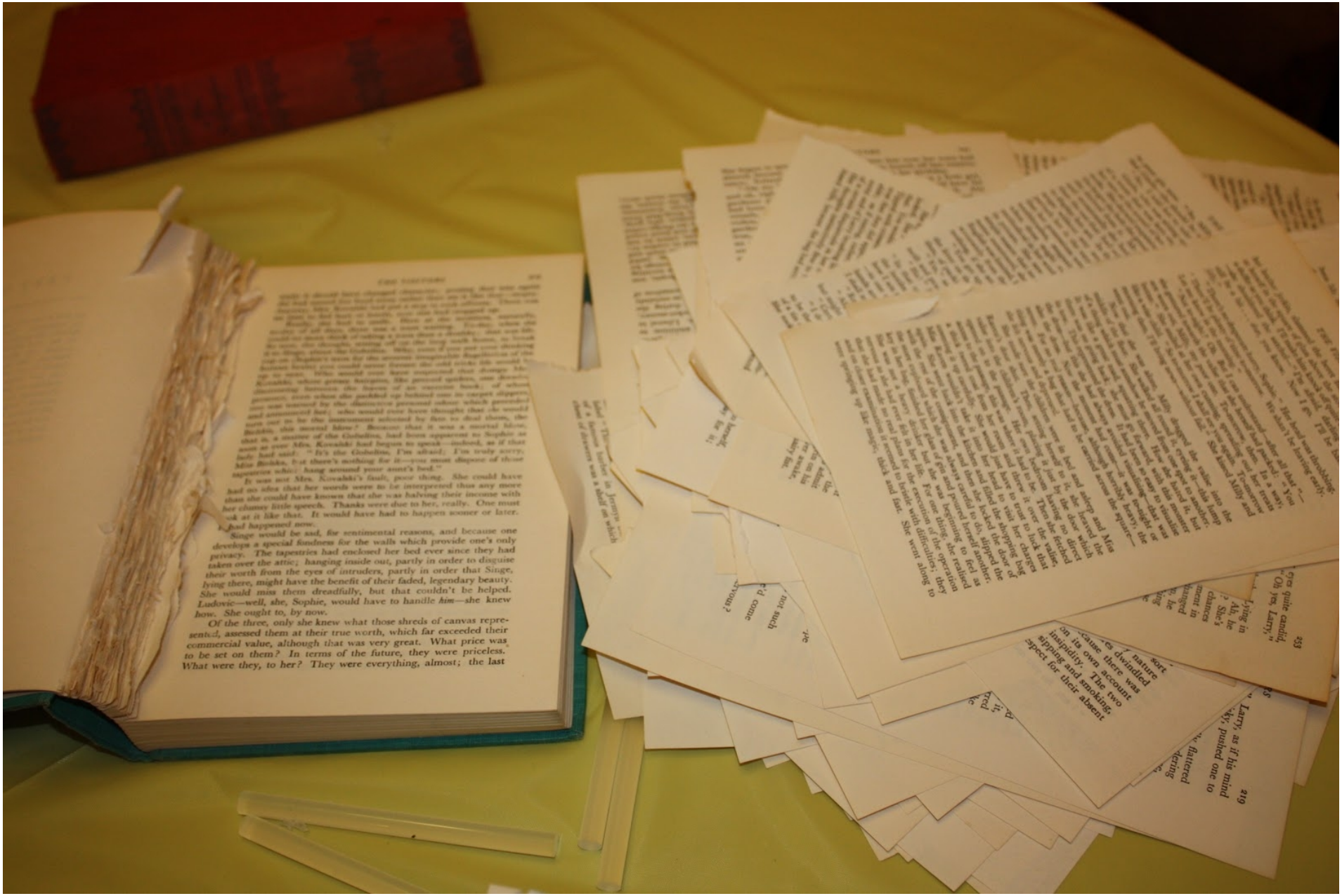
Two components to identify

- Where?
- How?
  - Finding that 'where'
  - Inserting the code

Where?

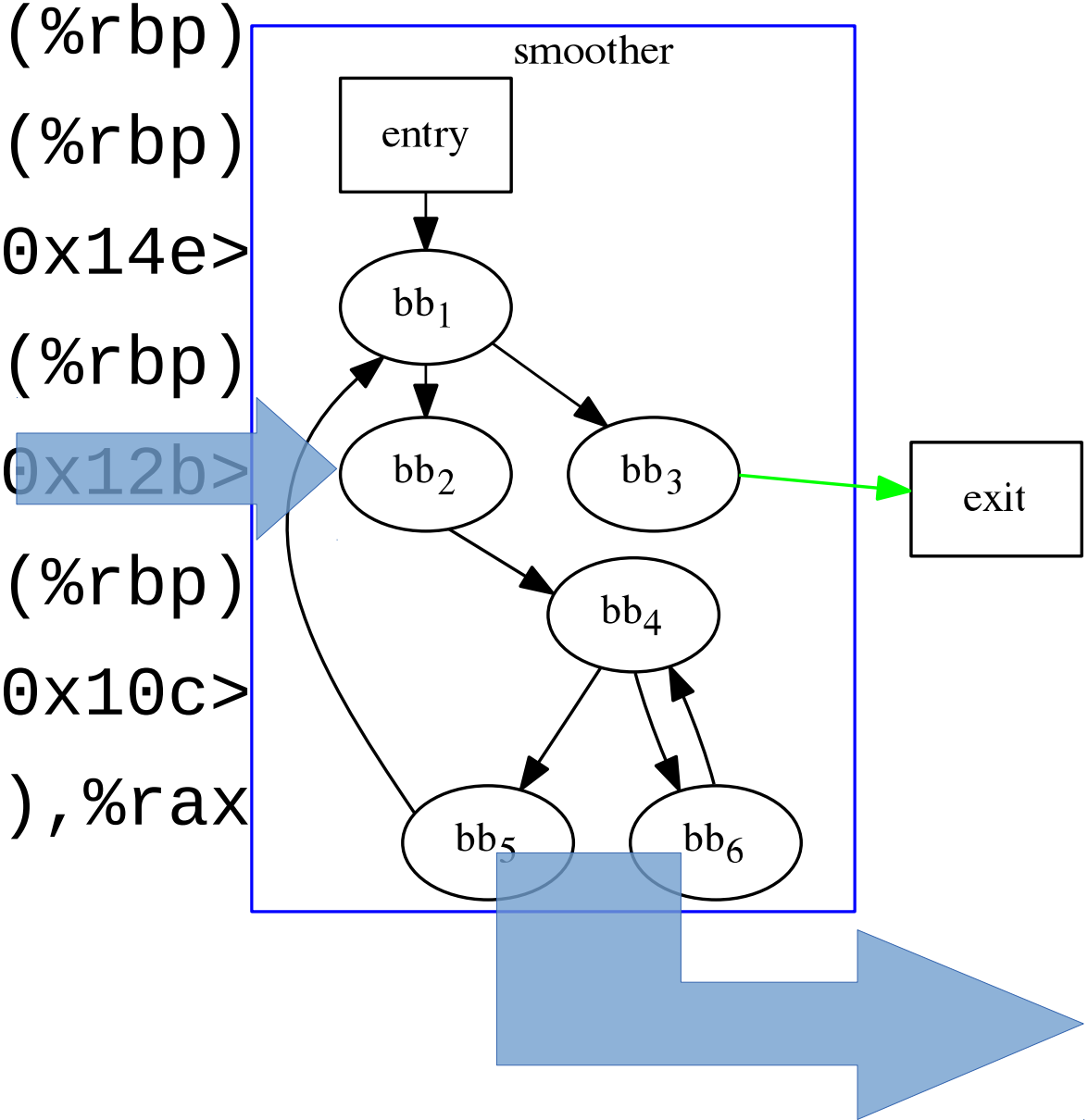
```
for(size_t y=0; y < dims[1]; ++y) {
  const size_t row=y*dims[0];
  for(size_t x=0; x < dims[0]; ++x)
    array[row+x] = …
}
vis(array, dims[0], dims[1], ...)
```

Page fault


Of the three, only she knew what those shreds of canvas represented, assessed them at their true worth, which far exceeded their commercial value, although that was very great. What price was to be set on them? In terms of the future, they were priceless. What were they, to her? They were everything, almost; the last

How: dynamic analysis

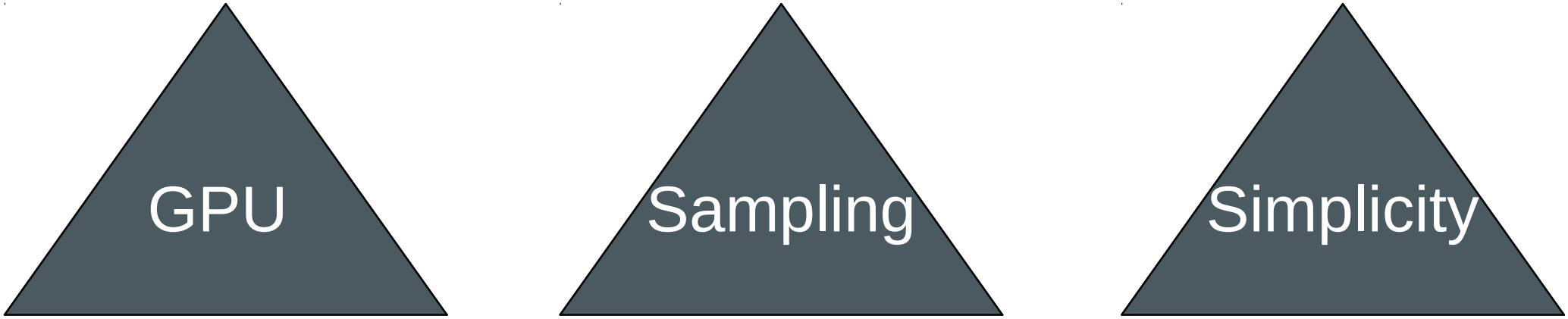
```
mov    %rax,-0x6b8(%rbp)
movq   $0x0,-0x6f0(%rbp)
jmpq   400bba <s3+0x14e>
movq   $0x0,-0x6e8(%rbp)
jmpq   400b97 <s3+0x12b>
movq   $0x0,-0x6e0(%rbp)
jmp    400b78 <s3+0x10c>
mov    -0x6f8(%rbp),%rax
add    $0x8,%rax
mov    (%rax),%rax
```



Decompilation

```
for(size_t y=0; y < dims[1]; ++y) {
  const size_t row=y*dims[0];
  for(size_t x=0; x < dims[0]; ++x)
    array[row+x] = …
}
```

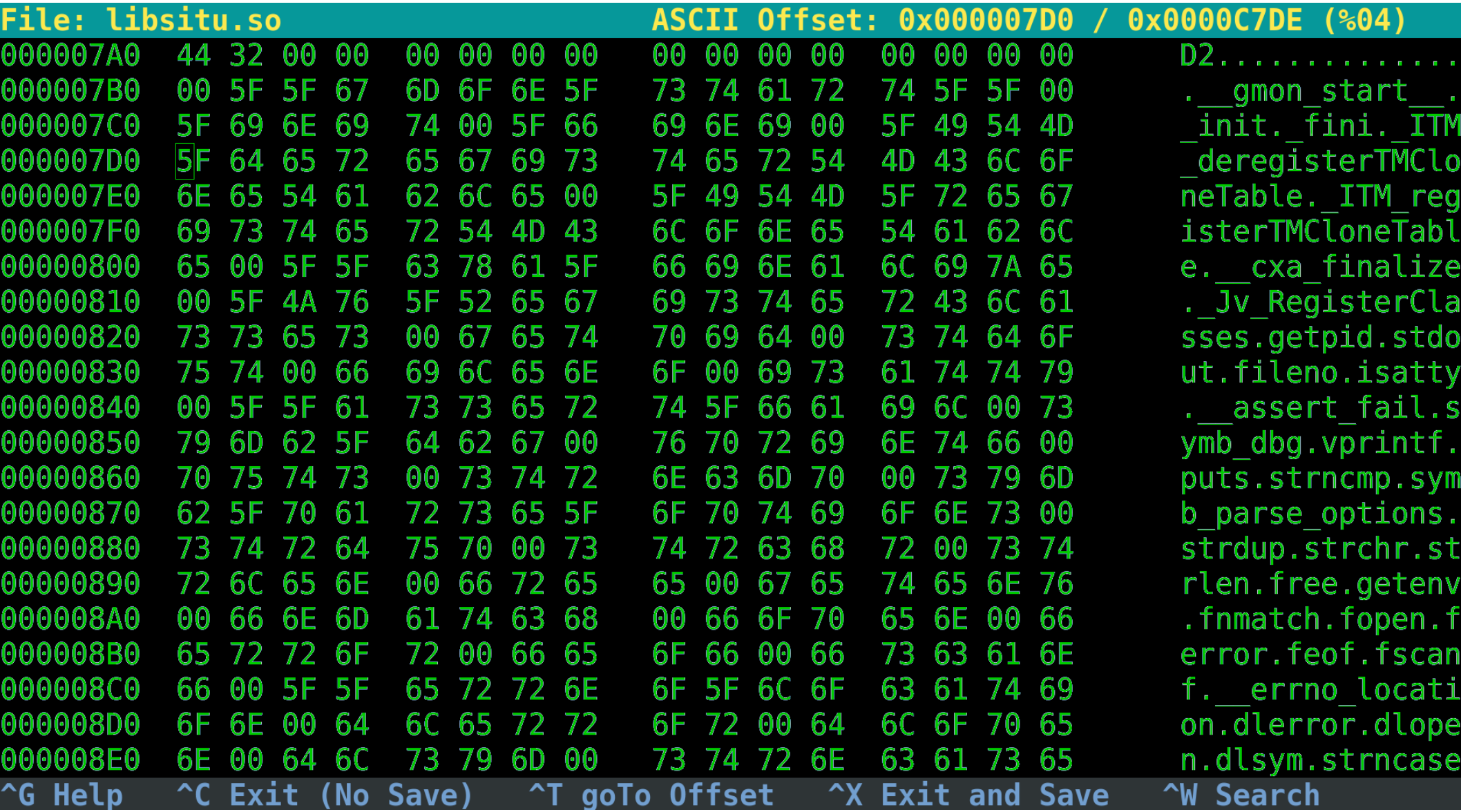
Summary



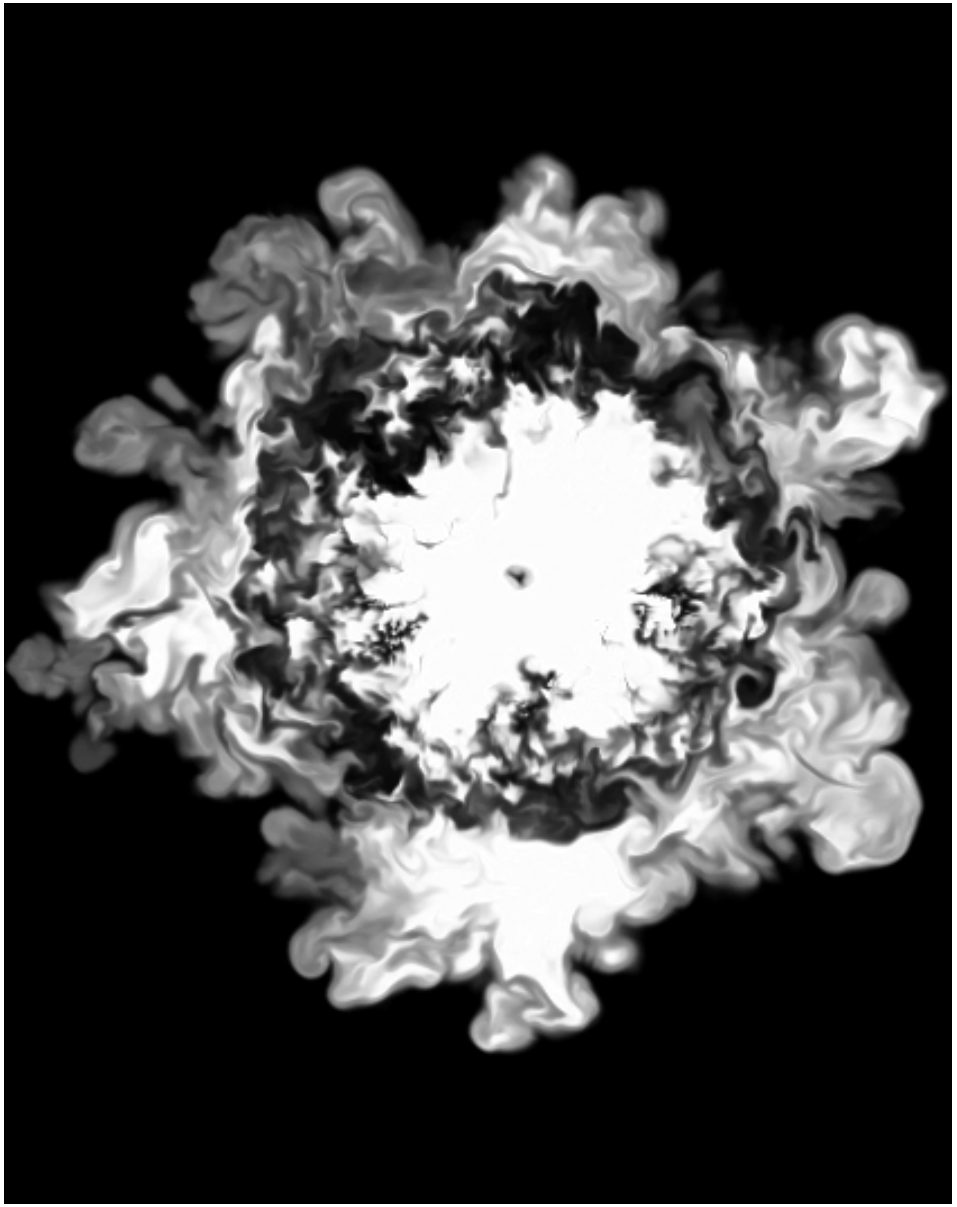
?

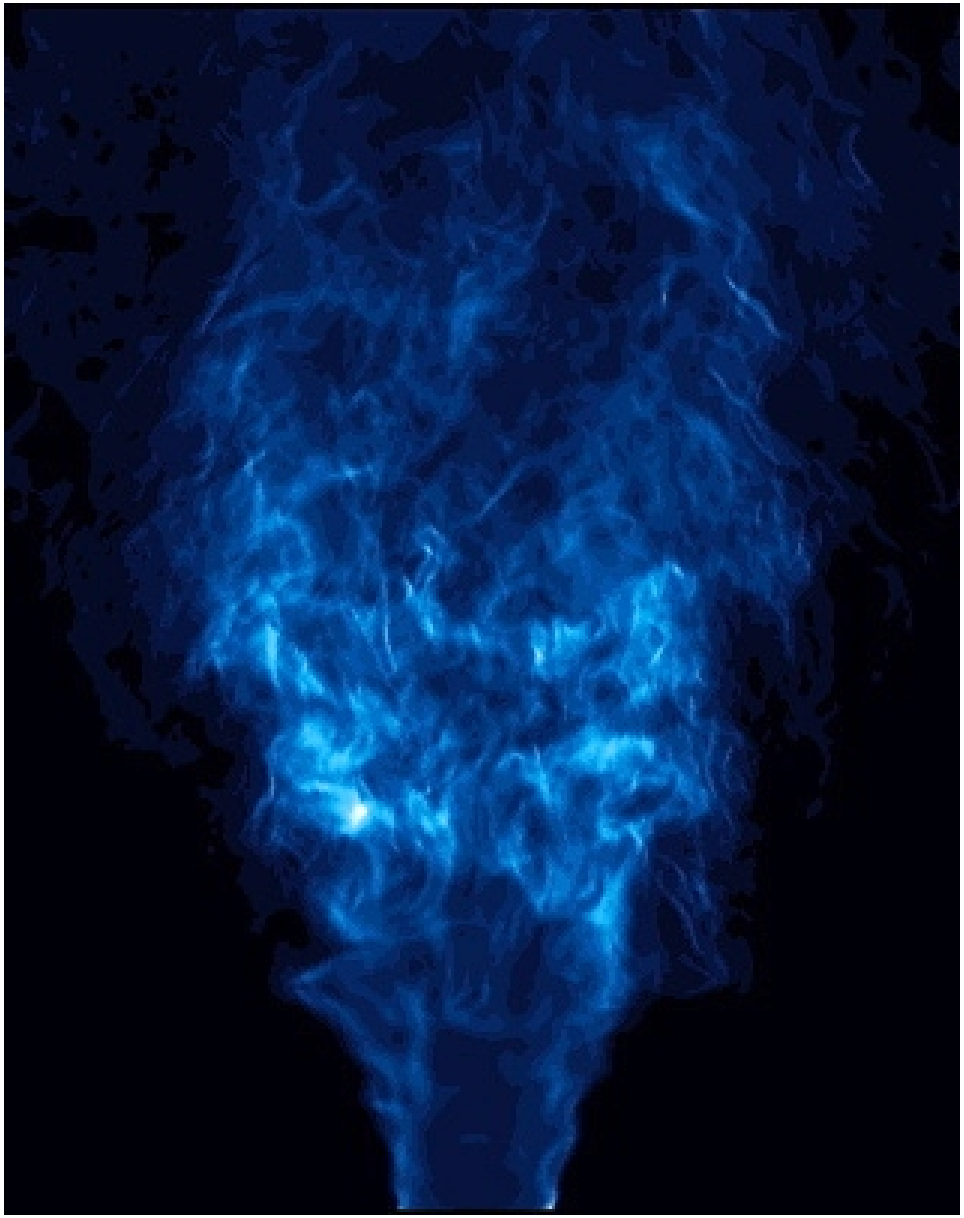
Thanks: Flickr User:jurvertson

Binary Instrumentation

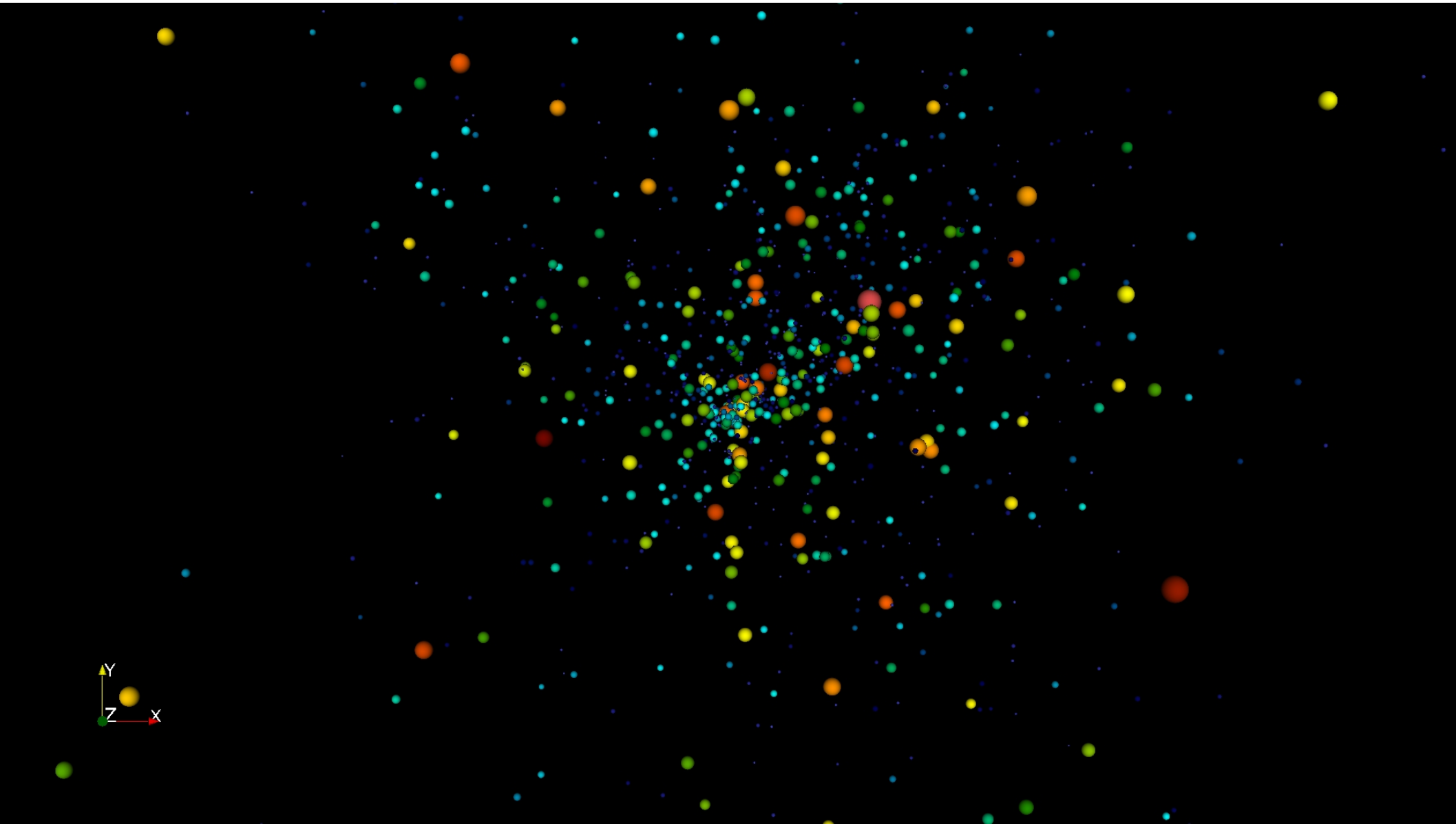


Traditional use cases are viable

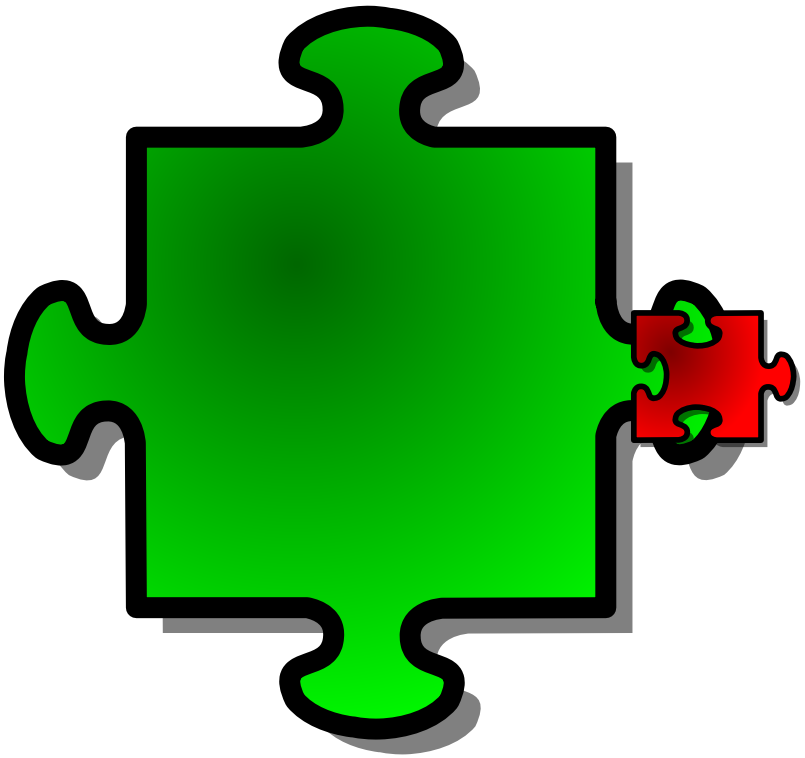




What *about* the data?



Small connectors encourage ad hoc computations



Network-based *in situ*

Image Credits

- Wikipedia User:Mysid, User:Lobsterbake, User:Someone35
- Flickr user jurvertson
- Tango project
- Silo web site
- My HPC students