methods of computational science

visualization
day ii - bottlenecks/parallel-viz

santiago v lombeyda
center for advanced computing research
caltech
quick review:
THE VISUALIZATION PROCESS
usual visualization “engine”

interaction

ui

data

processing algorithms
generation

rendering
data: geometric structure

abstract multi-dimensional data records

MPG Cylinders Horsepower Weight Acceleration Year Origin
8.50.42.88.2440.250.41500.45.30.469.582.54.83.23
18.000000 8.000000 130.000000 3504.000000 12.000000 70.000000 1.000000
15.000000 8.000000 165.000000 3693.000000 11.500000 70.000000 1.000000
18.000000 8.000000 150.000000 5436.000000 11.000000 70.000000 1.000000
16.000000 8.000000 150.000000 3433.000000 12.000000 70.000000 1.000000
17.000000 8.000000 140.000000 3449.000000 10.500000 70.000000 1.000000
...

2d/3d data + scalar/vector/tensor + time
# vtk DataFile Version 2.0
My Tet-Pyramid Example
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 4 float
0.0 0.0 0.0
1.0 0.0 0.0
0.5 0.0 0.7
0.5 0.6 0.7
CELLS 1 5
4 0 1 2 3
CELL_TYPES 1
10
POINT_DATA 4
SCALARS vertexData float 1
LOOKUP_TABLE default
0.1
0.2
0.3
0.4
CELL_DATA 1
SCALARS tetData int 1
LOOKUP_TABLE default
1
<?xml version="1.0"?>
<VTKFile type="ImageData" version="0.1"
  byte_order="LittleEndian">
  <ImageData WholeExtent="0 3 0 3 0 3" Origin="0 0 0"
    Spacing="1 1 1">
    <Piece Extent="0 3 0 3 0 3">
      <PointData Scalars="vertexData">
        <DataArray type="Float32" Name="scalarData"
          format="ascii">
          0 1 2 3 1 2 3 4 2 3 4 8 3 6 9 11
          2 3 4 5 5 6 7 8 3 4 5 6 4 5 6 7
          3 4 5 6 3 4 5 6 4 5 6 7 6 7 8 9
          2 3 4 5 2 3 4 5 3 4 5 6 4 5 6 7
        </DataArray>
      </PointData>

      <CellData Scalars="cellData" Normals="cell normals">
        <DataArray type="Int32" Name="cellData" format="ascii">
          1 3 9 2 8 1 6 3 9 2 7
          2 3 4 6 7 8 6 9 1 0
          0 1 2 0 2 4 1 2 3
        </DataArray>
      </CellData>
    </Piece>
  </ImageData>
</VTKFile>
#!/usr/bin/python
# load VTK extensions
import vtk

# create a rendering window and renderer
renderer = vtk.vtkRenderer()
myWindowRenderer = vtk.vtkRenderWindow()
myWindowRenderer.AddRenderer(renderer)
myWindowRenderer.SetSize(640,480)

myInteractiveWindow = vtk.vtkRenderWindowInteractor()
myInteractiveWindow.SetRenderWindow(myWindowRenderer)

# read mydata from volume.vti file
mydata = vtk.vtkXMLImageDataReader()
mydata.SetFileName("volume.vti")

# create filter
mydataIso = vtk.vtkMarchingCubes()
mydataIso.SetInput(mydata.GetOutput())
mydataIso.SetValue(0,0.1)
mydataIso.SetValue(1,0.3)
mydataIso.SetValue(2,0.5)
mydataIso.SetValue(3,0.7)

# pipe results to polymapper, and add actor
mydataMapper = vtk.vtkPolyDataMapper()
mydataMapper.SetInput(mydataIso.GetOutput())
mydataActor = vtk.vtkActor()
mydataActor.SetMapper(mydataMapper)

# assign our actor to the renderer
renderer.AddActor(mydataActor)

# enable user interface interactor
myInteractiveWindow.Initialize()
myInteractiveWindow.Start()
closer look:

BOTTLENECKS
visualization bottlenecks

- data size
- data format
- xml

data

processing algorithms

geometry generation

rendering

ui

interaction
visualization bottlenecks

- data
- processing algorithms
- geometry generation
- rendering
- interaction
- ui

- computation
- power
visualization bottlenecks

- number of triangles (base rendering units)
- number voxels

Data flow diagram:
- Data processing algorithms
- Geometry generation
- Rendering

Interaction with UI
visualization bottlenecks

- data
  - processing algorithms
  - geometry generation
  - rendering

- interaction

- knowledge level of “end user”
- complexity of data base
addressing: bottlenecks:

lod vs parallelism
addressing throughput bottlenecks

- level of detail (LOD)
  - requires preprocessing
  - requires larger storage (original+...)

- parallel processing/rendering
  - requires a parallel system
  - increases sw complexity
  - less likely to be “portable”
lod: decimation

25365

12681

50%

1011

4%
- quality
- algorithm <-> time
- when to use lod?
- improve interactivity
- smaller footprint
addressing: bottlenecks:
lod vs parallelism
addressing *throughput bottlenecks*

- level of detail (LOD)
  - requires preprocessing
  - requires larger storage (*original+...*)

- parallel processing/rendering
  - requires a parallel system
  - increases sw complexity
  - less likely to be “portable”
parallel viz...

- processing algorithms
- geometry generation
- rendering

data -> interaction

ui

+ cpus

data?

seams?
parallel viz...

- + gpus
- screen?
- space?
addressing: bottlenecks:
leverage other peoples work: parallelism
visualization system

- data
- processing algorithms
- geometry generation
- rendering
- interaction
- ui

- Paraview
- LLNL VisIt
- EnSight
- OpenDX
- IBM’s DataExplorer
- Mollegro, ...
tools: PARAVIEW VTK
paraview

- vtk based!
- active community
  - mailing list + wiki
  - lead at kitware: berk geveci
    - sandia national lab
    - los alamos national lab, army research lab
- parallel!
- QT based (from version 3.0)
  - 3.4.0 available in most platforms
quick demo:
PARAVIEW
tools:
VISIT  VTK
llnl’s visit!

* vtk based!
* active development
* very responsive
* lawrence-livermore national lab (asci/doe)
* designed for large data
* parallel!
* presets for large lab machines
* easily scriptable via python
* simple docs *(minimal pdf)*
demo: VISIT
why paraview\textsuperscript{PV}? (visit\textsuperscript{V!}?)

- paraview/visit can handle large data!
  - parallel
  - stable
  - gui
  - lod

- many filters (all of vtk), open source, binaries for major platforms, 64 bit versions, actively being developed, scriptable (python!), ...

- possible to extend and/or script
  - learning curve
  - hard to get everything
visualization bottlenecks

- Data
- Processing algorithms
- Geometry generation
- Rendering

Interaction

UI
case study: gui design
MCELL BIO VIEWER
intuitive
simple
interactive
effective

who is the user?
what is the task?

...inspiration
GUI

![GUI of iTunes](image)
GUI
GUI
GUI

...a first draft
visualization bottlenecks

what if there is not much to leverage?
create your own vtk data file
- have at least 100 data points/grid points
- if you have some ascii based data already
  - you can used awk/sed to filter data
  - then you can manually add the vtk tags

create a movie using paraview!
(or visit)

if you feel ambitious, try data explorer, or write a vtk program
thanks!

avyakta.caltech.edu:8888/esci101