

# HYDRA Progress Report: Vectorized HDF5 Module

Michael S. Noble

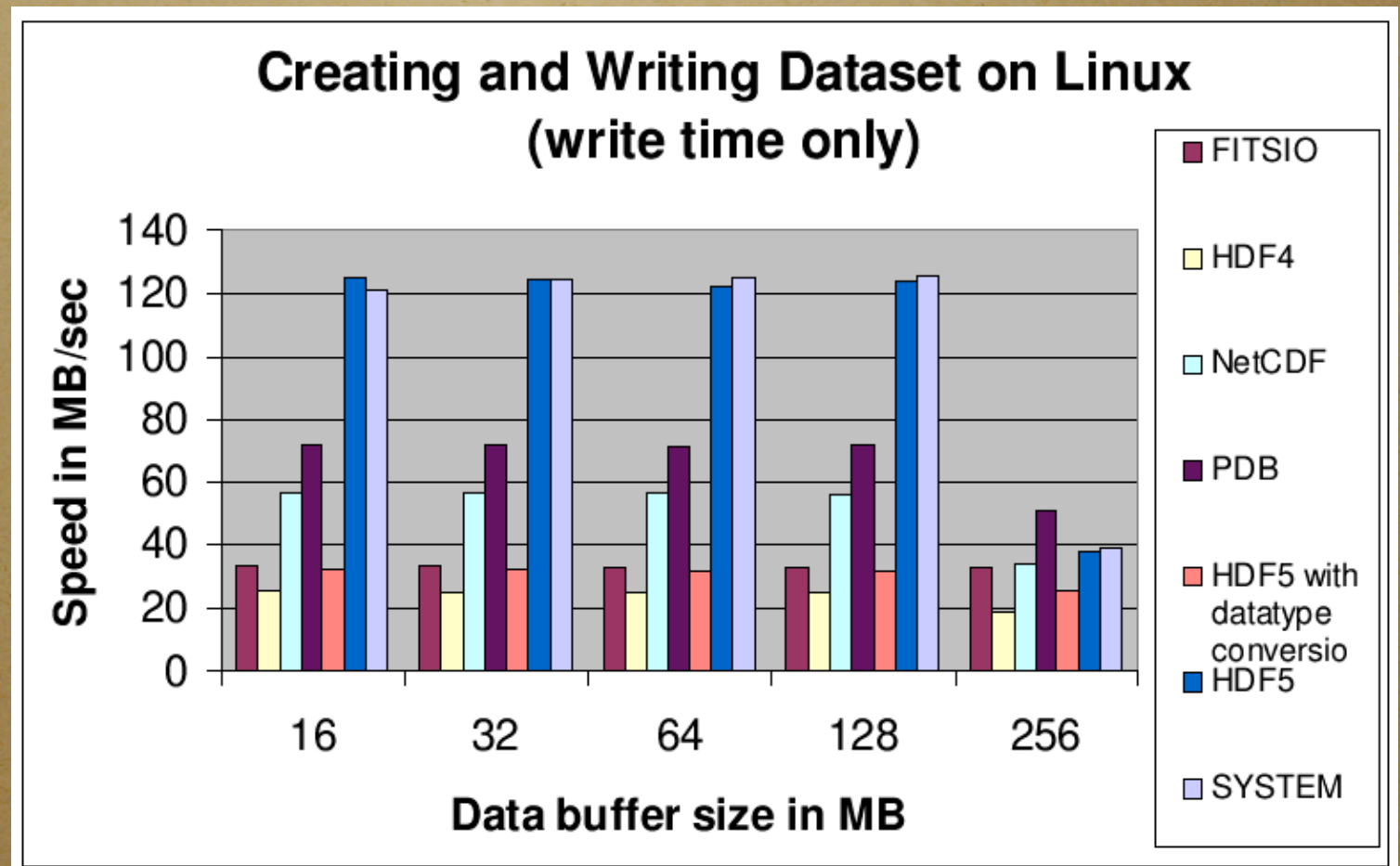
Kavli Institute for Astrophysics and Space Research  
Massachusetts Institute of Technology

June 29, 2006  
(updated 7/10/2006)

# HDF5: File Format & Software Library

Portable: no big/little endian concerns  
Scalable: Terabyte scale datasets  
Parallel: integrated MPI support

Very High  
Performance



Source: [hdf.ncsa.uiuc.edu/HDF5/RD100-2002/HDF5\\_Performance.pdf](http://hdf.ncsa.uiuc.edu/HDF5/RD100-2002/HDF5_Performance.pdf)

# HDF5: continued ...

Format of choice for large-scale science/engineering

NASA / EOS : Earth Observing System

FLASH: hydro simulations of thermonuclear flashes

many more ...

BUT: can require significant commitment for usage

Conceptually deep

Object management can be tediously verbose

Library contains **hundreds** of functions

# SLh5 Module

Boils HDF5 api down to 5 simple functions

h5_new	create one or more HDF5 files
h5_open	open one or more HDF5 files
h5_close	close one or more HDF5 files
h5_read	read one or more datasets/attributes
h5_write	write a dataset or attribute

Vectorized: multiple open/read operations per call

# SLh5: Clean and Simple

READ: no need to explicitly open/close files  
groups  
datasets  
attributes  
datatypes

WRITE: no need to explicitly create HDF5 types/groups

HDF5 type inferred from S-Lang type  
group creation inferred from dataset path

```
h5_write("out.h5", "/Group1/Group2/dataset", [1:500])
```

## RESULT

FLASH AMR i/o codesize 75% shorter than IDL(tm) version

IDL v6.1 (tm)

SLh5 v0.3.2

```
f = H5F_OPEN(file)
  d = H5D_OPEN(f, dataset)
    data = H5D_READ(d)
  H5D_CLOSE(d)
H5F_CLOSE(f)
```

```
data = h5_read(file, dataset)
```

```
f = H5F_OPEN(file)
  for i = 0, nvar-1 do begin
    d = H5D_OPEN(f, datasets[i])
      data = H5D_READ(d)
    H5D_CLOSE(d)
    array[i] = data
  endfor
H5F_CLOSE(f)
```

```
array = h5_read(file, datasets)
```

Input-Only  
Non-vectorized

Input OR Output  
Vectorized

Fosters casual use of HDF5 for astrophysical analysis

# SLh5: Fast

FLASH 3.x  
IDL 6.1 (tm)

```
idl> tstart = systime(1) &  
      read_amr, "jet_power_hdf5_plt_cnt_1866" &  
      print, systime(1) - tstart
```

```
18.238482
```

```
idl>
```

ISIS 1.3.3  
S-Lang 2.0.6  
SLh5 0.3.2

```
isis> tic; read_amr("jet_power_hdf5_plt_cnt_1866"); toc  
1.79431
```

```
isis>
```

391 Mb checkpoint file generated by FLASH

1.8 Ghz AMD Athlon, 2 GB RAM (Debian Sarge, GCC 3.3.5)

Approximately 1000% (10x) faster ???????

# SLh5: Fast

10x performance disparity seems too big, so let's just compare raw I/O on 7 individual datasets

IDL 6.1 (tm)

```
dsets = ["dens", "pres", "velx", "vely", "velz", "Bubb", "gpot" ]
fid = h5f_open("jet_power_hdf5_plt_cnt_1866")
tic
for i = 0, 6 do begin
    did = h5d_open(fid, dsets[i])
    dset = h5d_read(did)
endfor
print, 'IDL runtime: ', toc()
```

IDL Runtime: 2.52639

ISIS 1.3.3  
S-Lang 2.0.6  
SLh5 0.3.4

```
dsets = ["dens", "pres", "velx", "vely", "velz", "Bubb", "gpot" ]
tic ; data = h5_read("jet_power_hdf5_plt_cnt_1866", dsets)
vmmessage("SLh5 runtime: %S", toc)
```

SLh5 runtime: 1.71235

Closer now, but still ~50% performance advantage for SLh5

# Relevance to HYDRA

Conceptually: HYDRA offers 3 components

Source Models

Instrument Simulations

Fitting

(catch-all term for data filtering, iterative model adjustment, visualization)

## Source Model Types

Already Have: XSPEC - style 1D models  
Houck 3D cartesian grid model

Now Can Do: Hydrodynamics simulations  
(FLASH amr output)

# Example

3D simulation of shock passing over bubble  
(Sebastian Heinz, MIT, 2006)

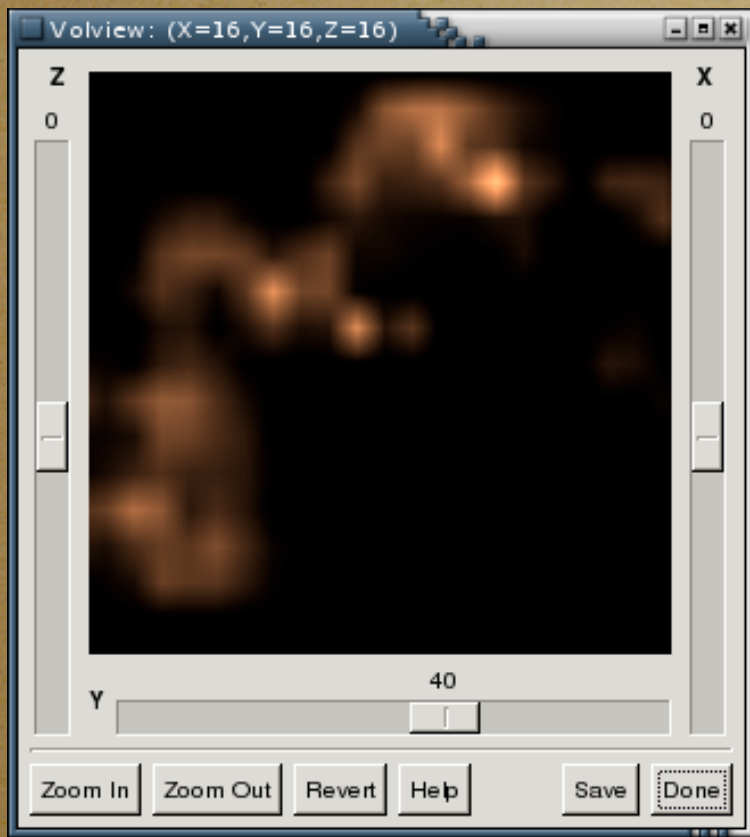
```
isis> (tree, params, unknowns) = read_amr("jet_power_hdf5_plt_cnt_1866")
```

```
isis> print(unknowns)
```

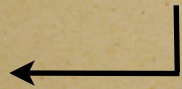
```
dens = Float_Type[3409, 16, 16, 16]  
pres = Float_Type[3409, 16, 16, 16]  
velx = Float_Type[3409, 16, 16, 16]  
vely = Float_Type[3409, 16, 16, 16]  
velz = Float_Type[3409, 16, 16, 16]  
Bubb = Float_Type[3409, 16, 16, 16]  
gpot = Float_Type[3409, 16, 16, 16]
```

# Poor Man's 400-Frame 3D Movie

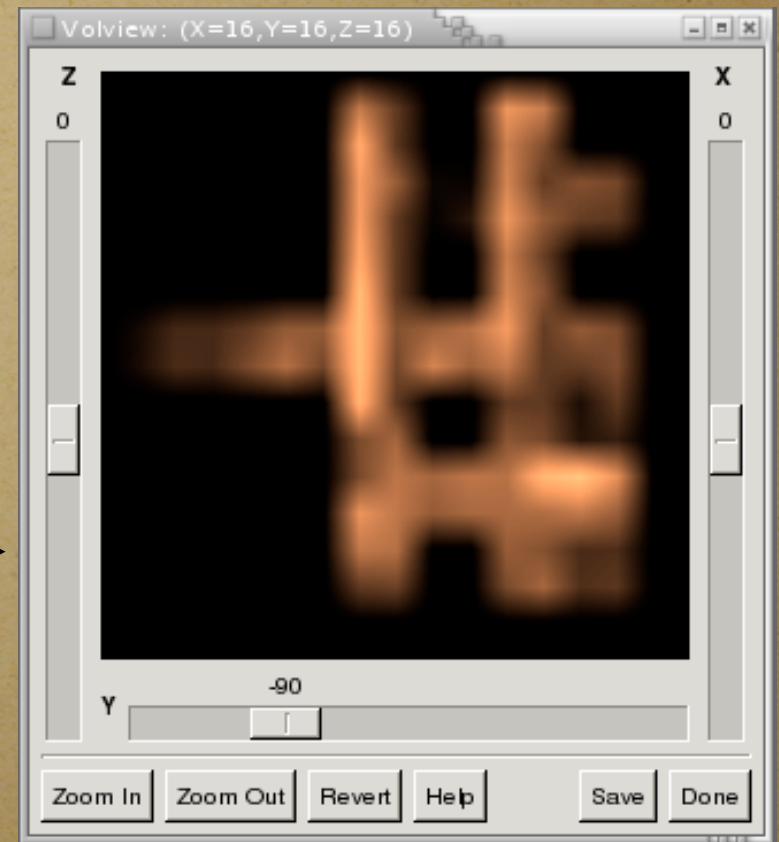
```
isis> _for i (0, 399, 1) volview(unknowns.Bubb[i,*,*,*])
```



frame  
zero



frame  
396



volview now also accepts 4D arrays, for stepping over series of 3D volumes