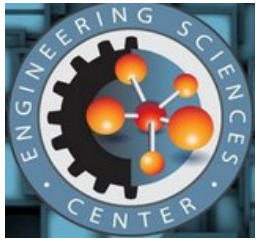
SPARTA Basics

Steve Plimpton
Sandia National Labs
sjplimp@sandia.gov

DSMC15 Short Course
Sept 2015 - Kapaa, Hawaii



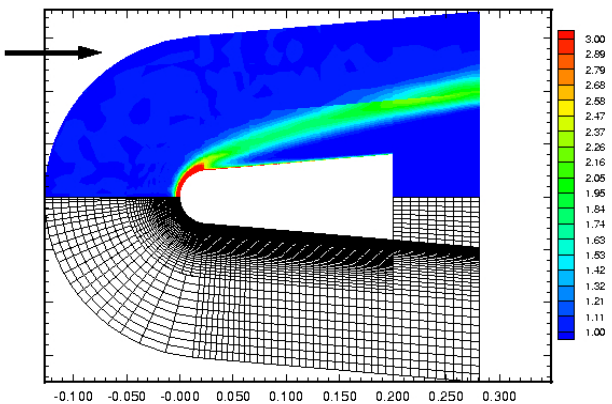
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

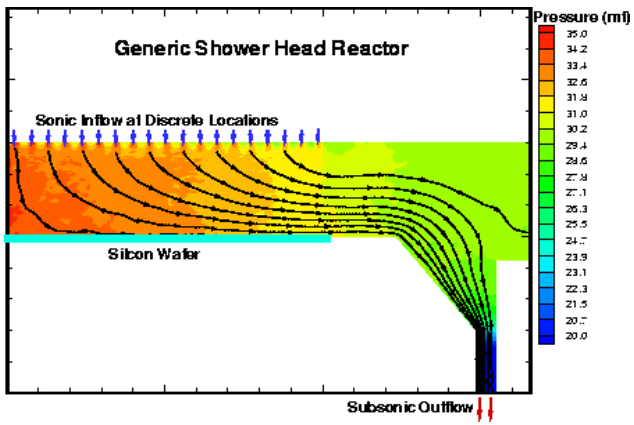
Outline

1. SPARTA and other DSMC codes
2. Getting started:
   - download, build, run an example, plot and viz
3. Basic input script options:
   - define simulation box and grid
   - create particles (species and mixtures)
   - define surfaces
   - turn on collisions & chemistry
   - gather statistics
   - perform output
4. Post-processing:
   - plotting stats
   - visualization

How SPARTA came about

- 1990s: Development of parallel ICARUS code at Sandia
  - started by Tim Bartel, parallelized by Steve
  - 2d, Fortran & MPI code, scaled to 1000s of procs
  - internal use only



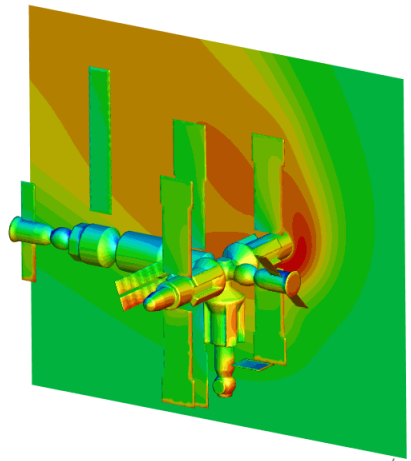


- 1998: Michael came to Sandia
  - added new physics modules to Icarus
  - used NASA DAC3D code to model space shuttle accident
  - worked with J Torczysnki & D Rader on slow-speed DSMC

How SPARTA came about, part II

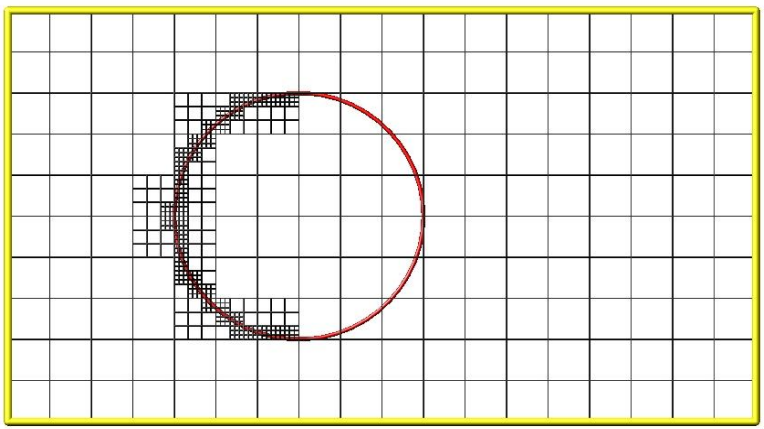
- 2011: DOE/NNSA funding for SPARTA
  - 3d, C++ & MPI
  - 3 main goals:
  - scalability on current/future HPC platforms:
    millions of cores, GPUs, Intel Phi, etc
  - easy to extend
  - open-source to enable:
    integration of user-contributed features
    external collaborations

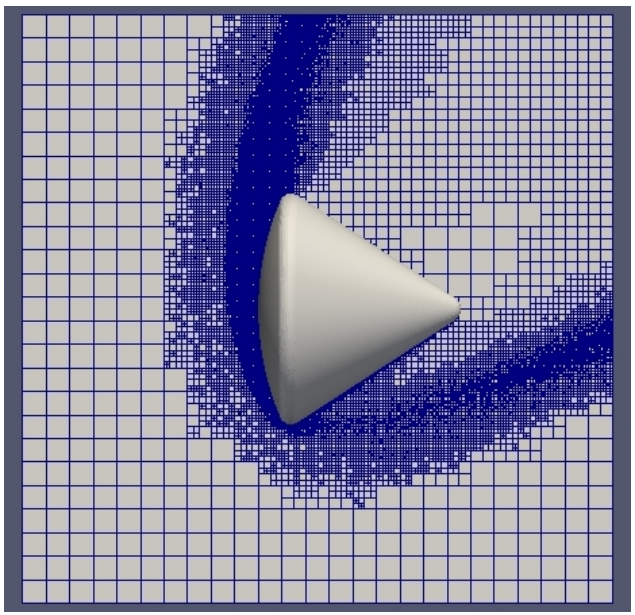
- July 2014: open-source release of SPARTA
  - http://sparta.sandia.gov
  - 950 downloads to date
  - updated continuously (no versions)
  - ∼40 updates in last year



DSMC 101, as implemented in SPARTA

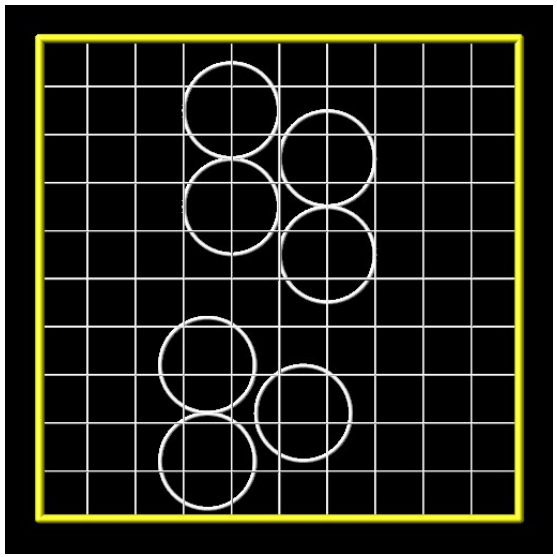
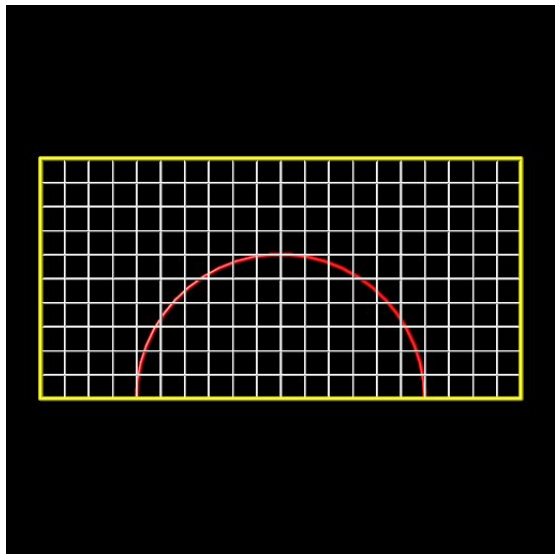
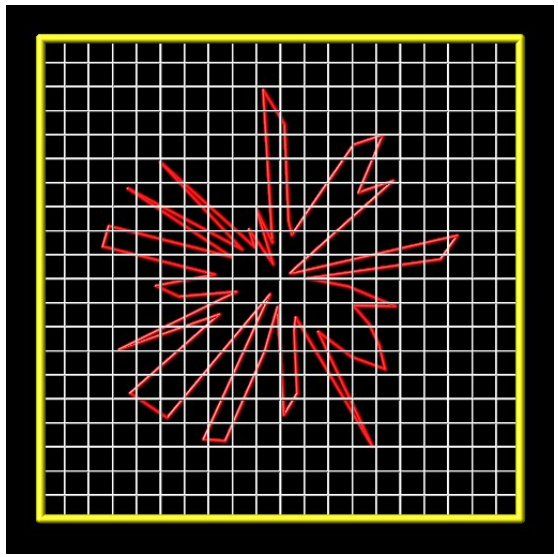
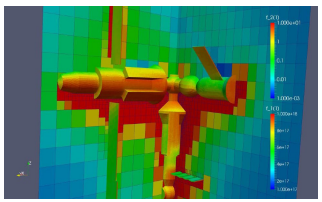
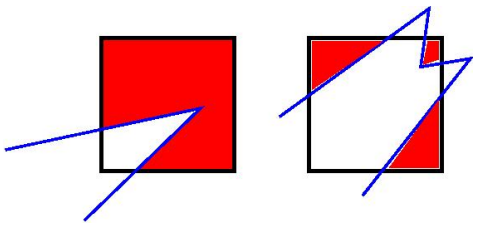
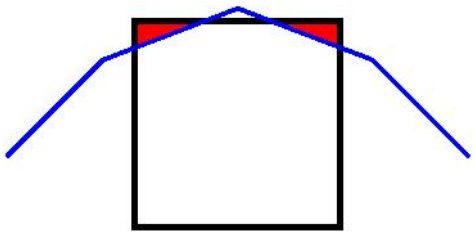
- Geometry:
  - 3d, 2d, axisymmetric (0d, 1d)
- Particles:
  - species, Fnum - same as any DSMC code
- Grid:
  - Cartesian (not body-fitted), hierarchical
  - uniform, 2 or 3 level, oct-tree, etc
  - levels only limited by fitting cell IDs in 64 bits





DSMC 101, as implemented in SPARTA

- Surfaces:
  - triangles (3d), line segments (2d or axisymmetric)
  - one or more watertight objects
  - objects can be clipped by simulation box
  - grid cells are cut/split by surface elements

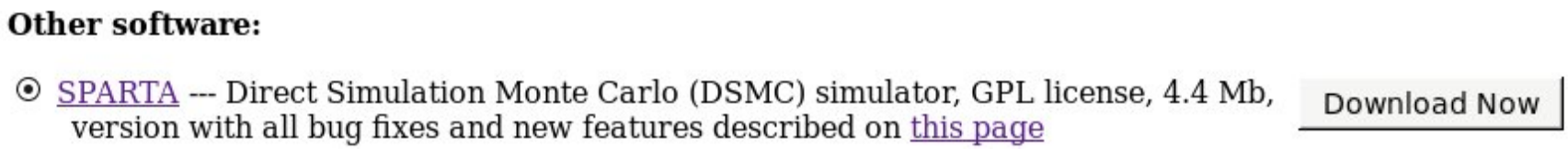







Other DSMC codes

- DS1V, DS2V, DS3V
  - original DSMC codes of Graeme Bird
- DAC3D
  - Lebeau group at NASA Johnson Space Center
  - 3-level Cartesian grids, shortcourse at DSMC11
- MONACO
  - Boyd group at U Michigan
- MGDS
  - Schwartzentruber group at U Minnesota
  - 3-level Cartesian grids
- SMILE
  - Russian group at Khristianovich Institute of TAM
- dsmcFOAM
  - group at U Strathclyde, Scotland
  - open-source, built on OpenFOAM CFD package
  - body-fitted finite-element style grids
  - shortcourse at DSMC13
- Other ?

Download SPARTA

- http://sparta.sandia.gov



- Unpack: tar zxvf sparta.tar.gz
- Should produce sparta-10Aug15 directory
  - README file
  - LICENSE = GNU General Public License (GPL)
  - bench = benchmark problems
  - data = files with species/reaction params, surface files
  - doc = documentation HTML file and PDF
  - examples = simple test problems
  - python = Python wrapper on SPARTA as a library
  - src = source files
  - tools = pre- and post-processing tools

Build SPARTA

% cd sparta-10Aug15/src
% make

```
make clean-all              delete all object files
make machine                build SPARTA where machine is one of:

# bgq = BlueGene Q, GNU compiler
# g++ = RedHat Linux box, g++4, MPICH2
# icc = RedHat Linux box, Intel icc, MPICH2
# mac = Apple PowerBook G4 laptop, c++, no MPI
# macmpi = Apple PowerBook G4, c++, OpenMPI
# mpi = RedHat Linux box, mpicc
# serial = RedHat Linux box, g++4, no MPI
# ...
```

% make serial
% make mpi

Should produce spa_serial or spa_mpi

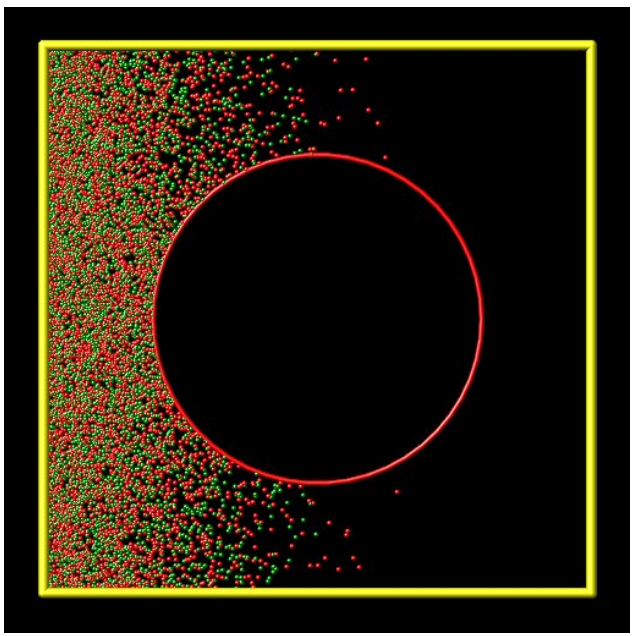
Make options

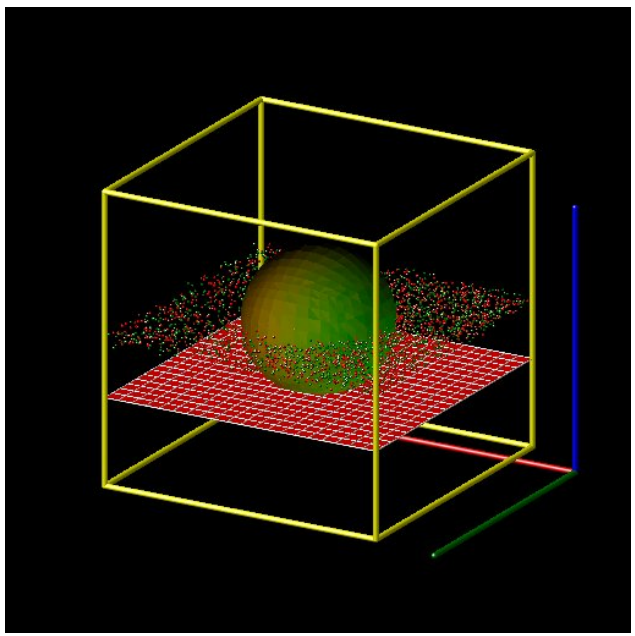
- Compiler settings
- Options:
  - MPI library
  - JPEG/PNG support
- doc/Section_start.html has details

```
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC =          g++
CCFLAGS =     -g -O # -Wall # -Wunused
SHFLAGS =     -fPIC
DEPFLAGS =    -M

LINK =        g++
LINKFLAGS =   -g -O
LIB =
SIZE =        size


# SPARTA ifdef settings, OPTIONAL
# see possible settings in doc/Section_start.html#2_2 (step 4)

SPARTA_INC = -DSPARTA_GZIP -DSPARTA_JPEG

# MPI library, REQUIRED
# see discussion in doc/Section_start.html#2_2 (step 5)

MPI_INC =       -DMPICH_SKIP_MPICXX
MPI_PATH =
MPI_LIB =     -lmpich -lmpl -lpthread

# JPEG and PNG library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 7)

JPG_INC =
JPG_PATH =
JPG_LIB =     -ljpeg
```

Run an example input script

- Dozens in examples sub-directories
  - see examples/README for one liners
- Copy SPARTA executable to working dir (not required)
  - % cd circle; cp ../../src/spa_serial .
- Redirect input script into executable (or -in)
- Run in serial or parallel:
  - % spa_serial < in.circle
  - % mpirun -np 4 spa_mpi < in.sphere





Demonstration

- I'll download, build, run an example problem,
  make a movie and plot, run benchmark in parallel
- Michael will demonstrate ParaView for interactive viz
- If you have SPARTA installed on your laptop,
  feel free to follow along ...

- Live demo, fingers crossed, no safety net ...

What's an input script

- Each line is a command
  - first word is name of command
  - required & optional arguments follow, separated by white-space
  - blank lines and # comments and & continuation allowed
- SPARTA executes each command as soon as it is read
  - does NOT read entire script, then perform a simulation
  - allows you to set params, run, change params, run, etc
- Every command has its own doc page:
  - Commands link on web page: http://sparta.sandia.gov
  - doc/Section_commands.html#cmd_5

adapt_grid	balance_grid	boundary	bound_modify	clear	collide
collide_modify	compute	create_box	create_grid	create_particles	dimension
dump	dump image	dump_modify	dump movie	echo	fix
global	group	if	include	jump	label
log	mixture	move_surf	next	partition	print
quit	react	read_grid	read_restart	read_surf	region
remove_surf	reset_timestep	restart	run	scale_particles	seed
shell	species	stats	stats_modify	stats_style	surf_collide
surf_react	surf_modify	timestep	uncompute	undump	unfix
units	variable	write_grid	write_surf	write_restart	

Structure of a typical input script

1. Global settings
   - dimension, global fnum, boundary, timestep commands
2. Define simulation box
   - create_box command
3. Define grid
   - create_grid or read_grid command
4. Define surface elements (optional)
   - read_surf command
5. Define species and particles
   - species & mixture commands
   - create_particles and fix emit commands
6. Choose collision and chemistry models (optional)
   - collide, collide_modify, react, surf_react commands
7. Tally statistics
   - compute and fix ave/time, ave/grid, ave/surf commands
8. Define outputs
   - stats command = one line every M steps
   - dump command for particles, grid cells, surf elements
9. Run command to perform a simulation for N steps
10. Rinse and repeat steps 7,8,9 if desired
    - equilibrate vs steady-state vs change params, etc

Example input script - first half

Look at examples/circle/in.circle - 25 non-blank lines

```
seed            12345
dimension       2
boundary        o r p
global          nrho 1.0 fnum 0.001
global          gridcut 0.0 comm/sort yes

create_box      0 10 0 10 -0.5 0.5
create_grid     20 20 1
balance_grid    rcb cell

species         air.species N O
mixture         air N O vstream 100.0 0 0
```

Example input script - second half

```
read_surf      data.circle
surf_collide   1 diffuse 300.0 0.0
surf_modify    all collide 1

collide        vss air air.vss
fix            in emit/face air xlo

timestep       0.0001

dump           ...
dump_modify    ...

stats          100
stats_style    step cpu np nattempt ncoll nscoll ...
run            1000
```

Make a movie

Uncomment these lines in examples/circle/in.circle:

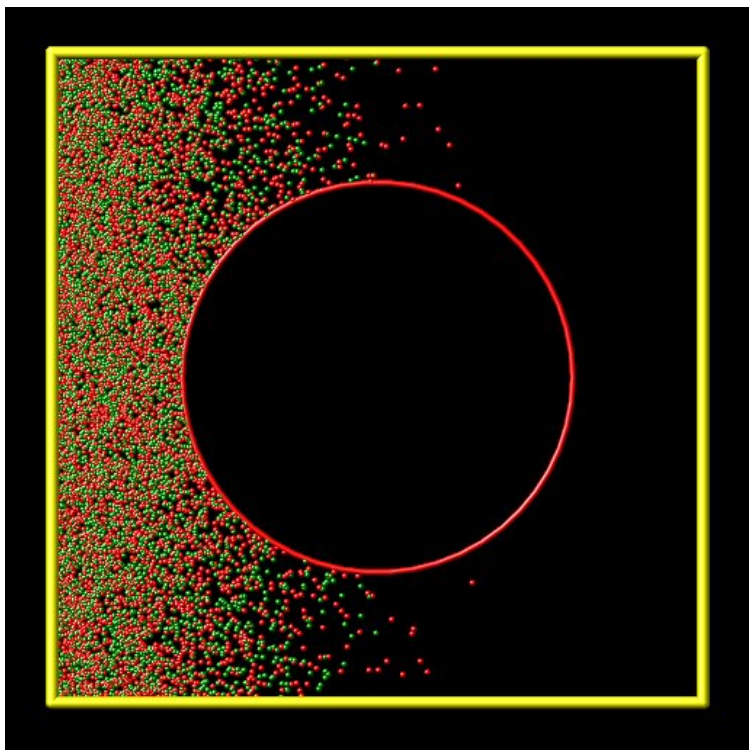
```
#dump          2 image all 50 image.*.jpg type type ...  &
#              surf proc 0.01 size 512 512 zoom 1.75
#dump_modify   2 pad 4
```

PPM, JPG, PNG files

- ImageMagick display
- Mac Preview

Create/view a movie

- ImageMagick
  convert *.jpg image.gif
- Open in browser
  open -a Safari image.gif
- Mac QuickTime
- Windows Media Player



Example output in log file

See log.sparta after running input script, same as screen output
Logs all input script commands and their output (if any)

```
create_grid 20 20 1
Created 400 child grid cells
  parent cells = 1
  CPU time = 0.00106001 secs
  create/ghost percent = 93.8596 6.14035
```

Example output - reading surface elements

```
read_surf data.circle
  50 points
  50 lines
  2 8 xlo xhi
  2.00592 7.99408 ylo yhi
  0 0 zlo zhi
  0.376743 min line length
  48 = cells with surfs
  104 = total surfs in all grid cells
  3 = max surfs in one grid cell
  0.753486 = min surf-size/cell-size ratio
  264 88 48 = cells outside/inside/overlap surfs
  48 = surf cells with 1,2,etc splits
  71.8 71.8 = cell-wise and global flow volume
```

Example output - memory usage and run stats

```
Memory usage per proc in Mbytes:
  particles (ave,min,max) = 0 0 0
  grid (ave,min,max) = 1.51388 1.51388 1.51388
  surf (ave,min,max) = 0.00348091 0.003479 0.00348282
  total (ave,min,max) = 1.51736 1.51736 1.51736

Step CPU Np Natt Ncoll Nscoll Nscheck
  0 0 0 0 0 0 0
  100 0.083772898 19773 0 0 123 4204
  200 0.20021796 31724 0 0 186 6740
  ...
  1000 1.281064 44115 0 0 212 8640
Loop time of 1.2811 on 4 procs for 1000 steps
  with 44115 particles
```

Example output - aggregate simulation stats

```
Particle moves = 37007526 (37M)
Cells touched = 41853734 (41.9M)
Particle comms = 141343 (0.141M)
Boundary collides = 172954 (0.173M)
Boundary exits = 166564 (0.167M)
SurfColl checks = 7313440 (7.31M)
SurfColl occurs = 172907 (0.173M)
Surf reactions = 0 (0K)
Collide attempts = 0 (0K)
Collide occurs = 0 (0K)
Reactions = 0 (0K)

Particle-moves/CPUsec/proc:  7.22183e+06
Particle-moves/step:  37007.5
...
```

Example output - timing breakdown & per-processor stats

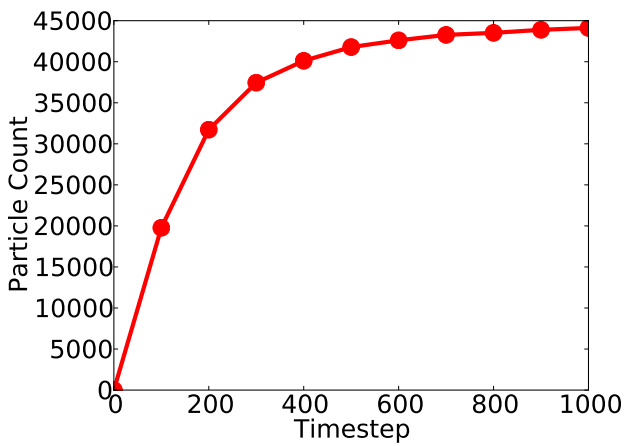
```
Move time (%) = 0.531144 (41.46)
Coll time (%) = 0.0237256 (1.85197)
Sort time (%) = 0.0362098 (2.82646)
Comm time (%) = 0.00583237 (0.455263)
Outpt time (%) = 0.617375 (48.191)
Other time (%) = 0.066813 (5.21528)
```

```
Particles:  11028.8 ave 17164 max 4879 min
Histogram:  2 0 0 0 0 0 0 0 0 2
Cells:  100 ave 100 max 100 min
Histogram:  4 0 0 0 0 0 0 0 0 0
GhostCell:  21 ave 21 max 21 min
Histogram:  4 0 0 0 0 0 0 0 0 0
EmptyCell:  21 ave 21 max 21 min
Histogram:  4 0 0 0 0 0 0 0 0 0
```

Plotting stats output from log file

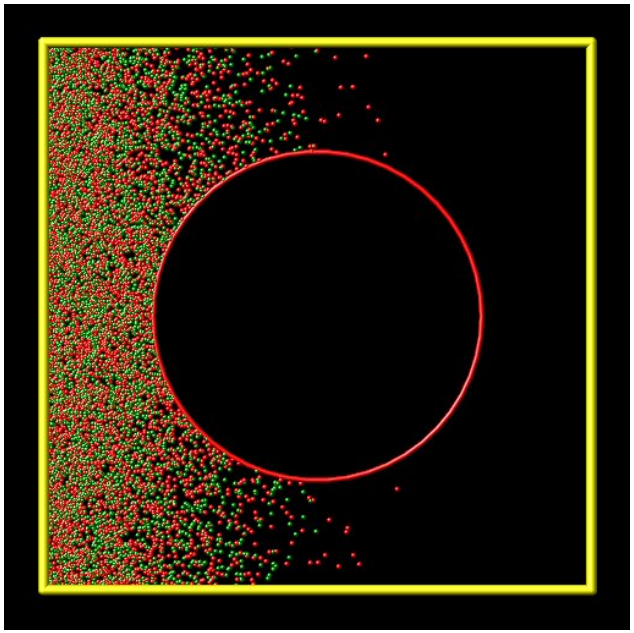
- tools/log2txt.py tool extracts stats columns from log file
- Discards non-stats info, concatenates multiple runs
- Example:
  python log2txt.py
  log.sparta data.txt
- tools/logplot.py tool extracts stats & wraps GnuPlot
- Example:
  python logplot.py
  log.sparta data.txt Step Np

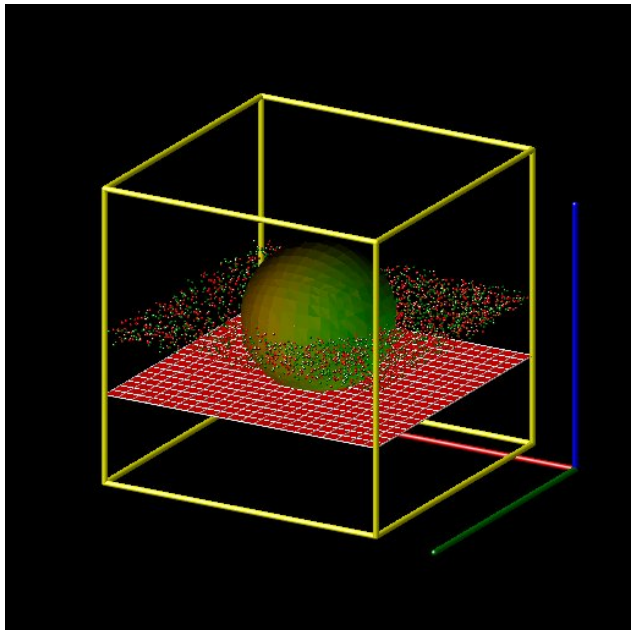
```
0.0 0.0 0.0 0.0 0.0 0.0 0.0
100.0 0.083772898 19773.0 0.0 0.0 123.0 4204.0
200.0 0.20021796 31724.0 0.0 0.0 186.0 6740.0
300.0 0.32452703 37448.0 0.0 0.0 198.0 7654.0
400.0 0.45519805 40114.0 0.0 0.0 175.0 7851.0
500.0 0.58969092 41785.0 0.0 0.0 189.0 8021.0
600.0 0.73009801 42606.0 0.0 0.0 174.0 8464.0
700.0 0.86730599 43271.0 0.0 0.0 210.0 8701.0
800.0 1.00442 43517.0 0.0 0.0 163.0 8315.0
900.0 1.141727 43888.0 0.0 0.0 190.0 8358.0
1000.0 1.281064 44115.0 0.0 0.0 212.0 8640.0
```



On-the-fly viz via dump image/movie commands

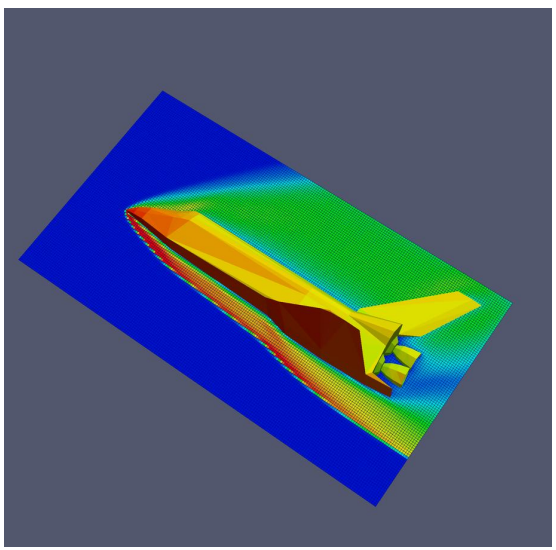
- Dump image and dump_modify commands
- Options for particles, grid cells, grid lines, surface elements, 2d slices, regions, etc
- Color by species, grid stats, surface stats, processor, etc
- Can change viewpoint dynamically $\Rightarrow$ fly-by movie
- Dump movie command creates movie file via FFmpeg
- Doc pages have all the details

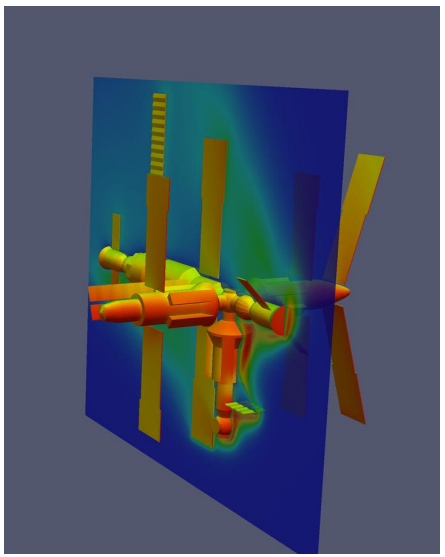


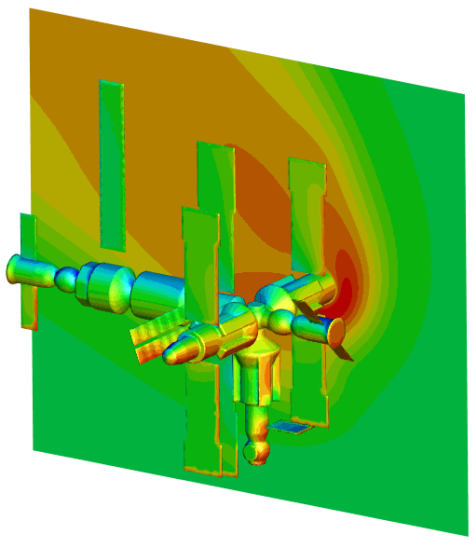


Visualization via Paraview or TecPlot

- Traditional interactive post-processing viz
- ParaView: http://www.paraview.org, freely available
- Tecplot: http://www.tecplot.com, commercial (free demo)
- Two tools to convert SPARTA output to ParaView input:
  - Python scripts in tools directory
  - grid data via grid2paraview.py
  - surface data via surf2paraview.py







Resources for learning SPARTA

Either on web site or in downloaded tarball:

- Tutorial (these slides)
- Manual: doc/Manual.html or doc/Manual.pdf
  - Intro, Commands, Howto, Modifying, Errors sections
- Alphabetized command list: one doc page per command
  - doc/Section_commands.html#cmd_5
- Examples: ~12 sub-dirs, ~25 input scripts
  - many have movies: http://sparta.sandia.gov/pictures.html
- Questions:
  - send email to Steve or Michael
  - plan to create a mail list at some point