Profiling and scalability testing for Beatbox

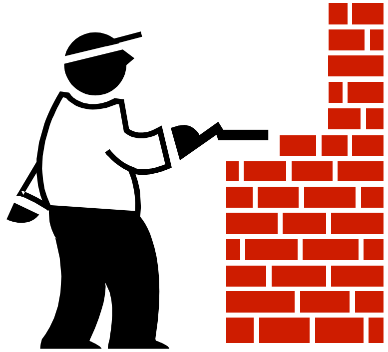
Mario Antonioletti
EPCC
mario@epcc.ed.ac.uk
+44 131 650 5141

Outline

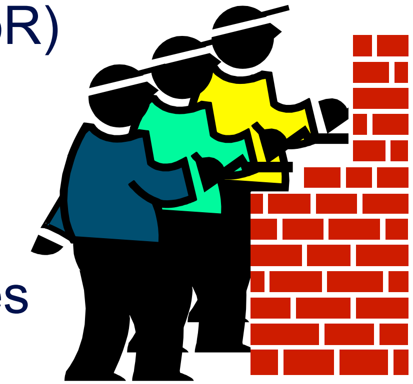
- Why parallelism?
- Some background
- Performance metrics
- Methodology
  - Scalability curves
  - Profiling
    - Example output
- Results
  - Rabbit ventricle, human atrium, box3D
- Conclusions

Why parallelism?

- It takes one brick layer 3 days to lay a wall
  - How long will it take 3 brick layers?
  - Opportunity to do things faster or bigger



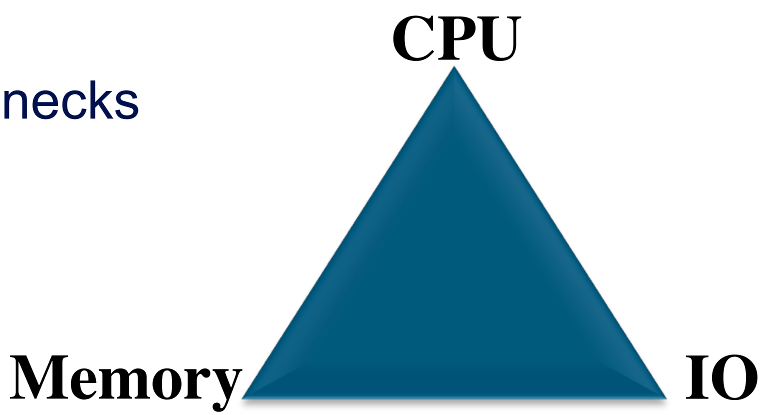
- Can use multi-core systems in the same way
  - Most laptops now come with multiple core systems
  - Can take advantage of computers on a network
    - Communications latencies may prove expensive
  - Can used dedicated parallel machines (e.g. HECToR)
    - Have fast communication interconnects



- Main parallelisation strategies:
  - OpenMP (multi-threading) shared memory machines
  - MPI explicit message passing
  - Can use both
- Beatbox uses MPI

Some background

- Beatbox scripts are agnostic as to whether they are:
  - Run serially
  - Run in parallel
- Beatbox is currently not memory or I/O constrained.
  - Issues more to do with obtaining enough CPU power
  - Impacts on the parallelisation strategy used
    - Domain decomposition used
- Need to determine how well the parallel code works
  - See how well it scales
  - Dive down to identify performance bottlenecks



Performance metrics: speed-up & efficiency

- Speed-up S_n:

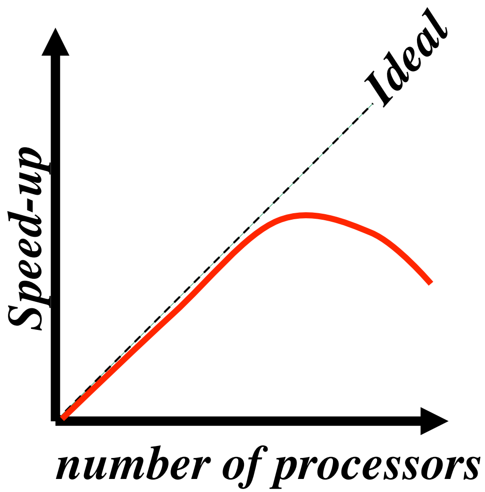
$$S_n = T_1 / T_n$$

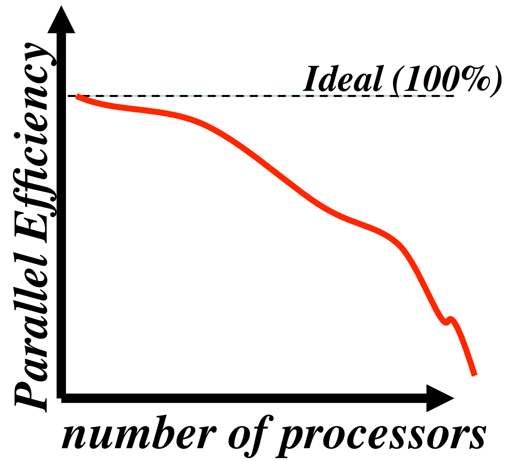
- Where:
  - T_1 is the execution time on 1 processor
  - T_n is the execution time on n processors

- Parallel Efficiency E_n:

$$E_n = S_n / n$$

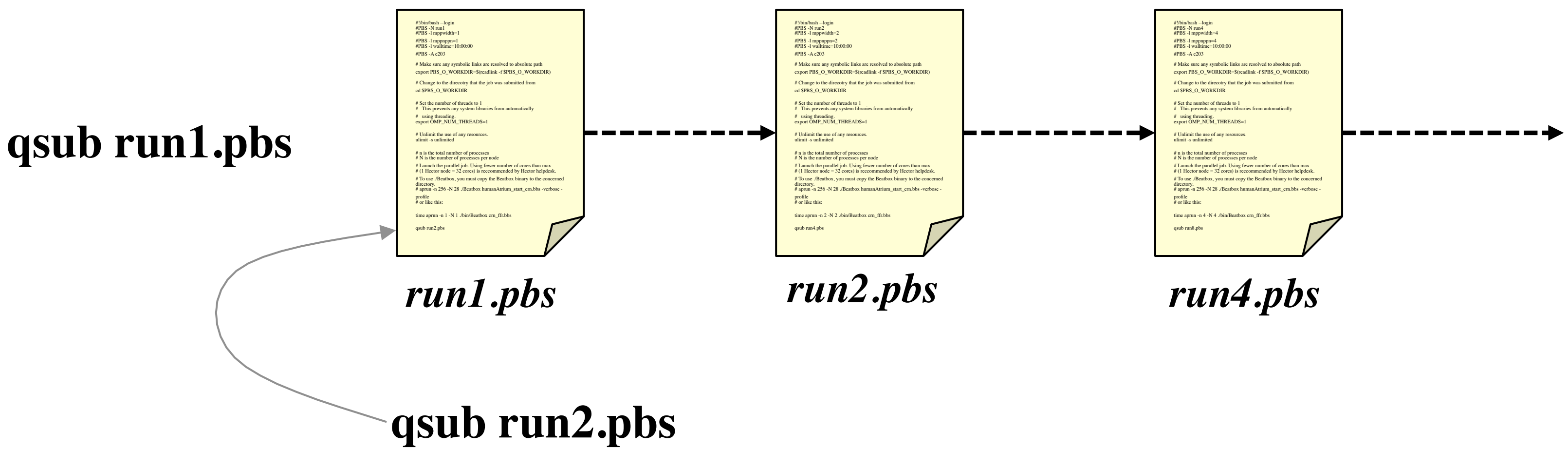
- Can also:
  - Strong scaling: fixed problem size throughout
  - Weak scaling: fixed problem size per processor





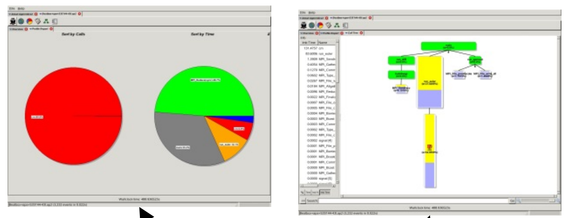
Methodology: scalability curves

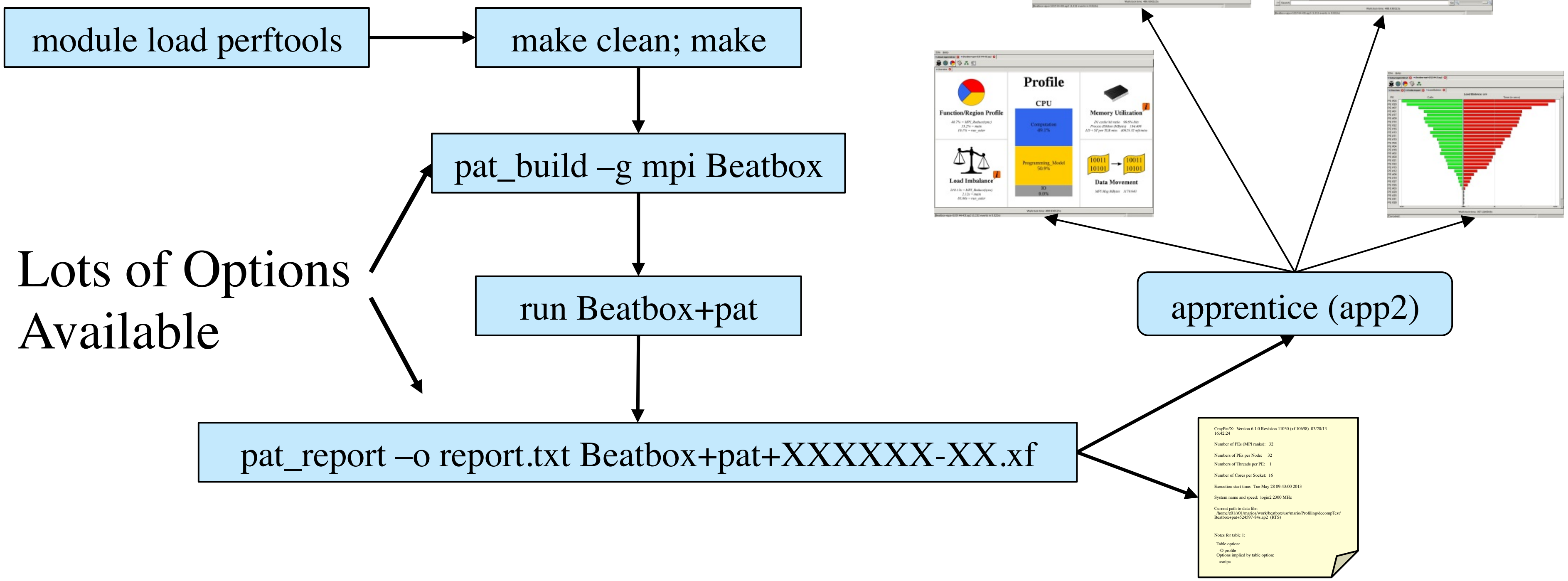
- Use chained PBS (Portable Batch System) scripts
  - PBS is the scheduling/batch system that operates on HECToR
- Could use shell script loops but max run time is 12 hours
  - Total run time for all the scripts can exceed that
- Variance not high so run jobs only once



Methodology: profiling

- Instrument the code to find out where it is spending time
  – Identify bottlenecks
- Cray Performance Analysis Tools (PAT)
  – Instrument executable
    – Perform sampling experiments
    – Perform tracing experiments





Profiling: example output

Load Imbalance
(max Time – Avg Time)

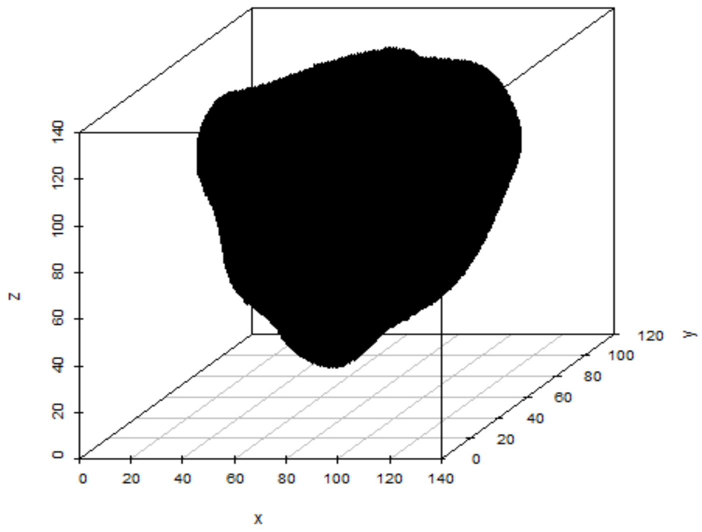
Where the code
is spending its time

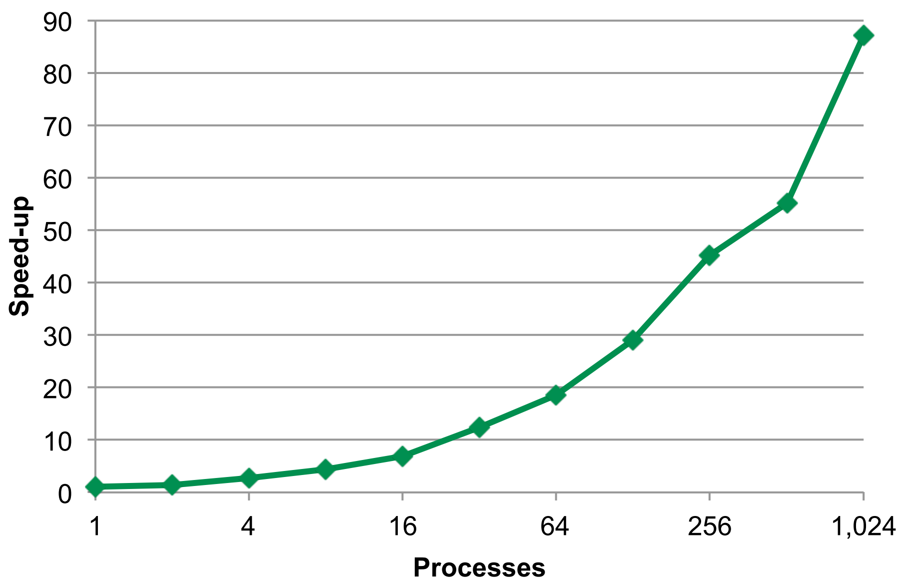
```
Table 1:  Profile by Function Group and Func

 Samp%    |    Samp   |  Imb.    |  Imb.    |Group
          |           |  Samp    | Samp%    | Function
          |           |          |          |  PE=HIDE

 100.0% | 26362.7 |      -- |      -- |Total
|-----------------------------------------------------------------
|  39.8% | 10499.8 |      -- |      -- |ETC
||----------------------------------------------------------------
||  27.1% |  7157.4 |  103.6 |   1.5% |__isoc99_vsscanf
||   4.8% |  1278.0 |   64.0 |   4.9% |____strtod_l_internal
||   3.0% |   790.1 | 1436.9 |  66.6% | cray2_EXP_14
||   2.4% |   640.3 |   59.7 |   8.8% |____strtol_l_internal
||   0.7% |   194.2 |   24.8 |  11.7% |_IO_getline_info
||   0.5% |   131.6 |  247.4 |  67.4% |_ALOG_15
||   0.2% |    60.1 |   13.9 |  19.4% |_IO_old_init
||   0.2% |    55.8 |   11.2 |  17.3% |_IO_str_init_static_internal
||   0.2% |    55.3 |   16.7 |  23.9% |_IO_no_init
||   0.2% |    46.4 |    9.6 |  17.7% |__isoc99_sscanf
||   0.2% |    41.3 |   21.7 |  35.5% |_IO_setb
||   0.1% |    26.0 |   68.0 |  74.7% |_EXP
…
```

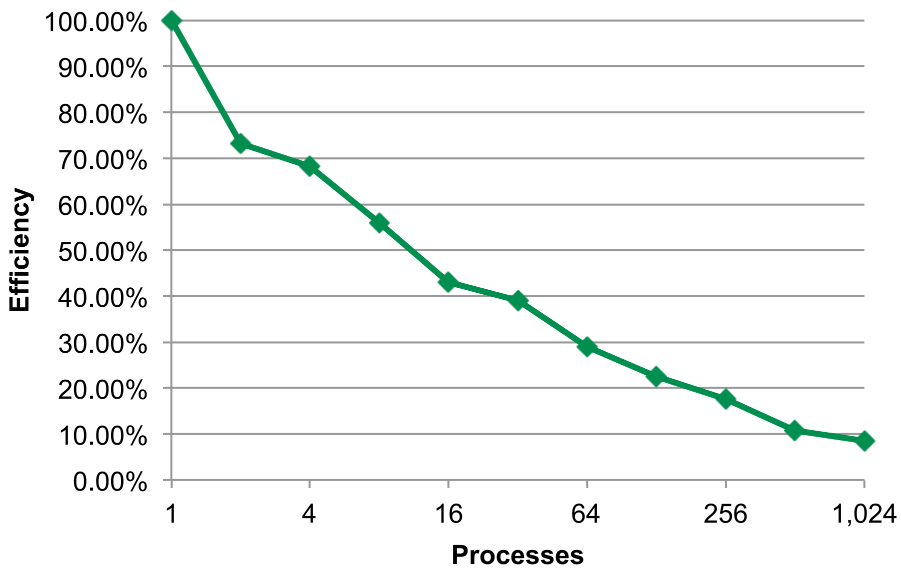
- Identify expensive parts
  - See if performance can be improved
- Caveat: don't want to optimise just one code execution path
  - Use different configurations/data files

Result: rabbit ventricle – FHN model

- Approximately 470k points
  - No output, 800 time steps
  - T_1 ~ 8900s, 11s per time step
- FHN model has 2 ODEs/cell

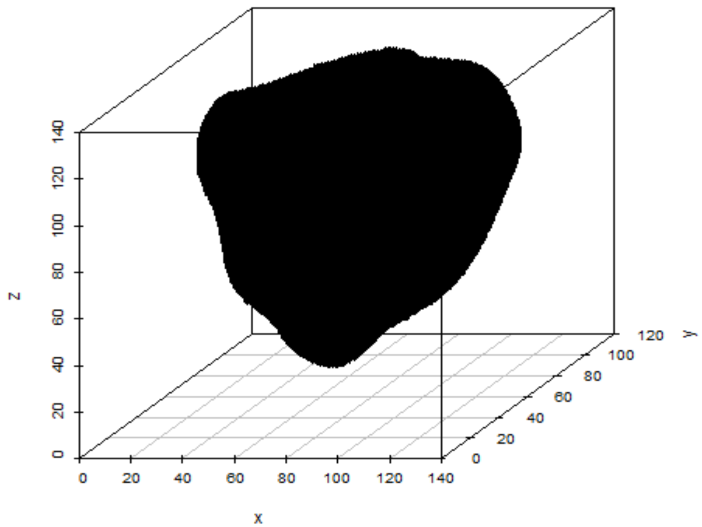


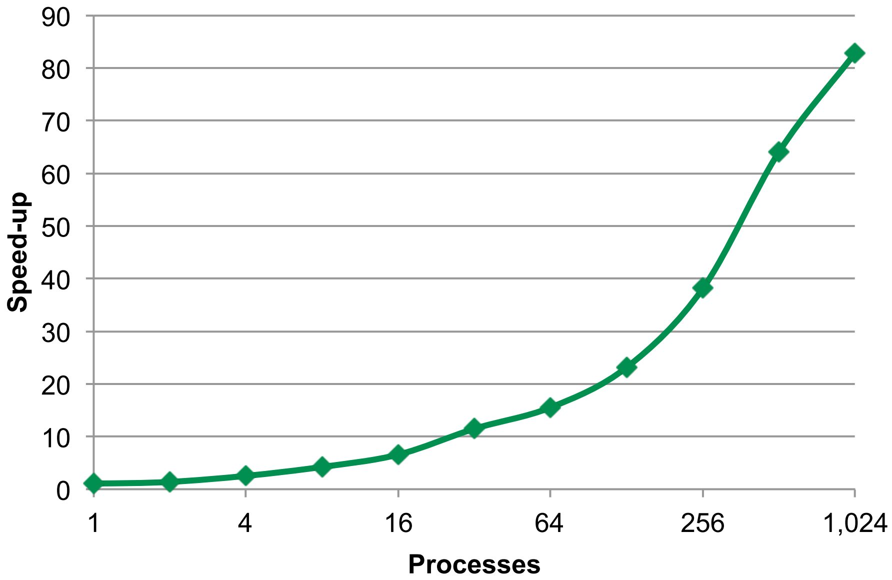


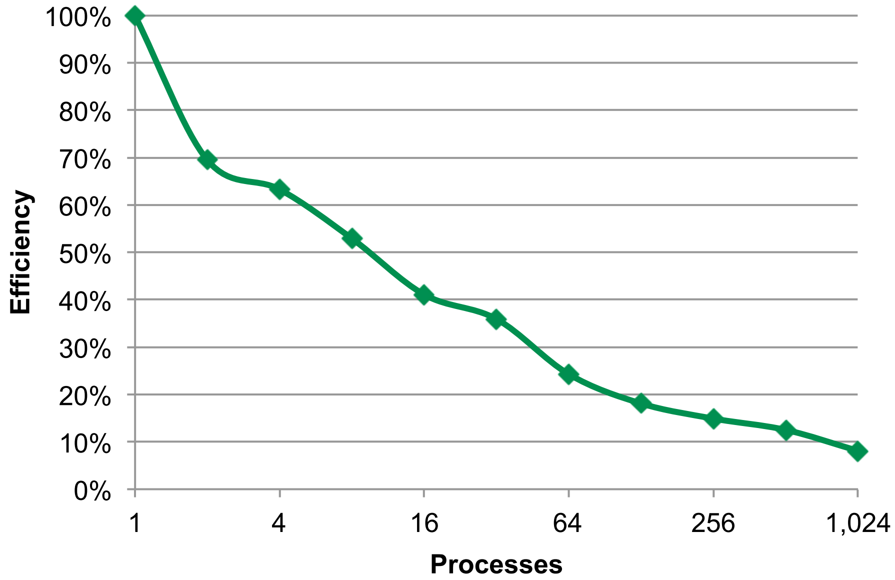


Result: rabbit ventricle – CRN model

- Approximately 470k points
  - No output, 10,000 time steps
  - T_1 ~ 12,273, ~1.2s per time step
- CRN model has 22 ODEs/cell

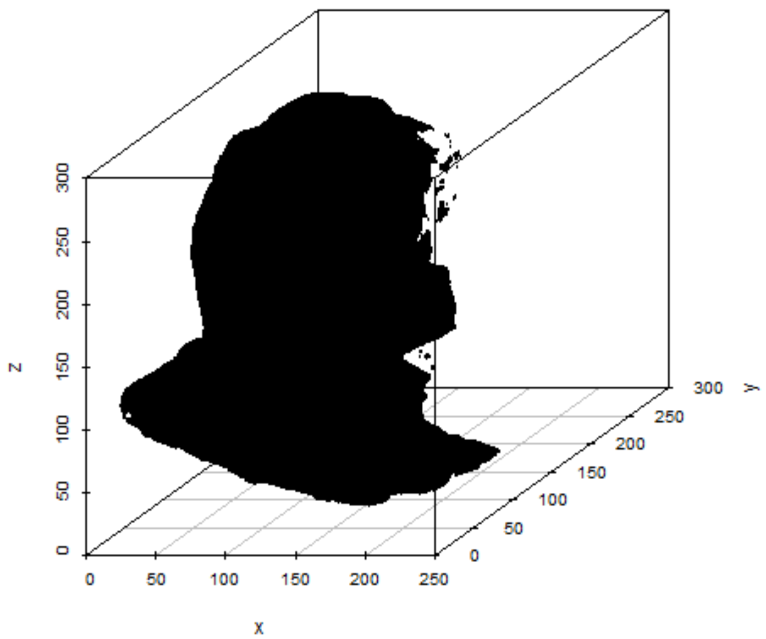


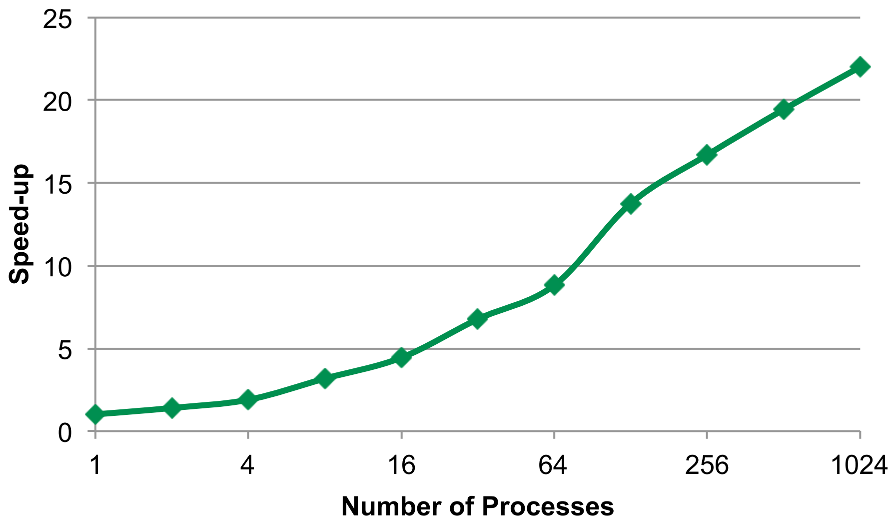


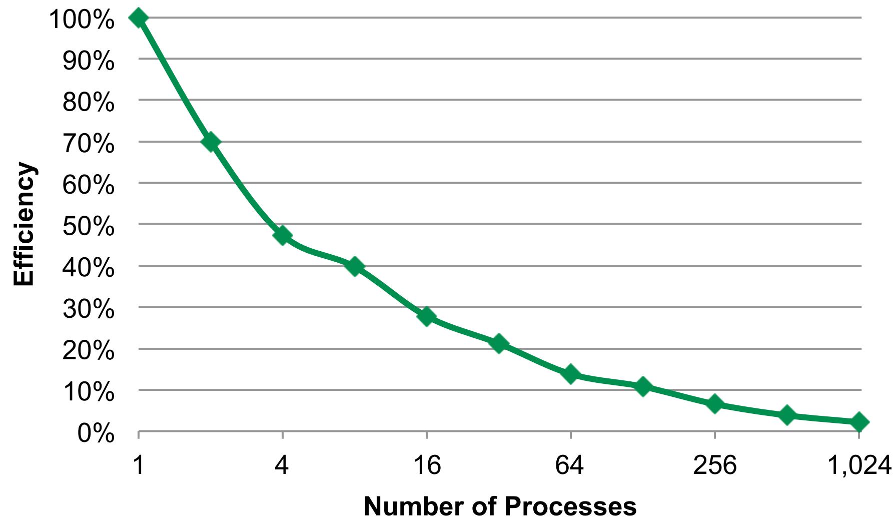


Result: human atrium

- Approximately 19M points
  - No output, 2000 time steps
  - T_1 ~ 5359, ~2.7s per time step
    - Compiled with –O3

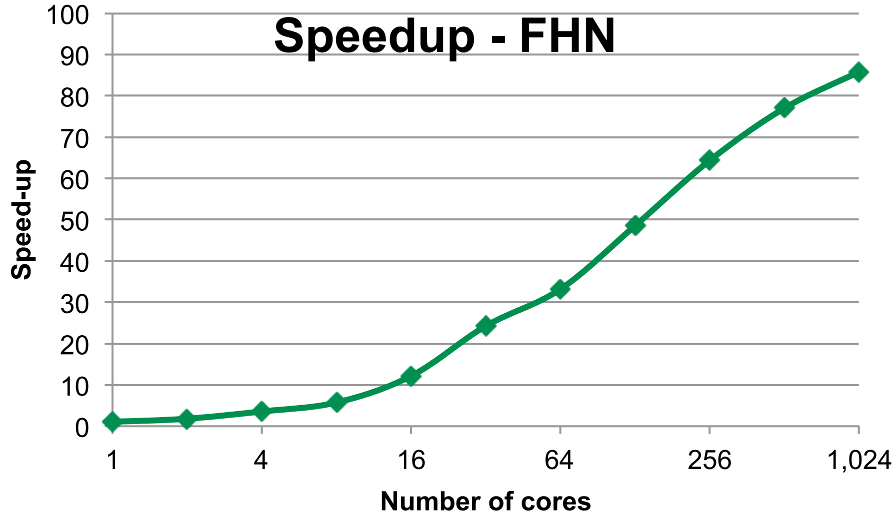


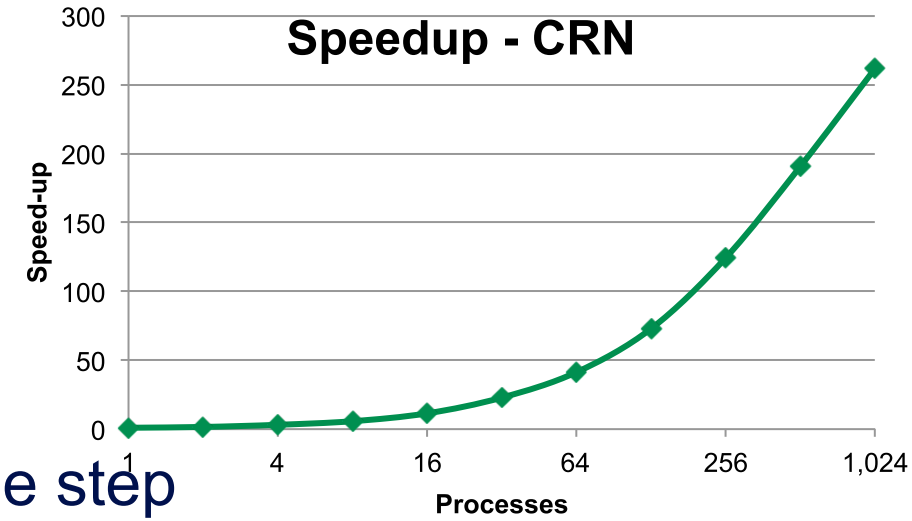




Result: Box3D

- Big box with biophysical realistic models
- Have a 302x302x302 grid
- FHN has 2 ODEs/cell
- CRN has 22 ODEs/cell
- No output
- FHN: 800 time steps
- FHN T_1 ~ 3430s, 4.8s per time step
- CRN:200 time steps
- CRN T_1 ~ 13,859s, 69s per time step





Conclusions

- Performance depends on:
  - The model used
  - How much fill there is
  - Performance quickly saturates as more processes are added
- You will get a definite benefit from using more processors
  - Do not have to go to HPC systems to observe this
  - Normally you want to achieve a performance of about 70%
- Need to identify where parallel performance bottlenecks are

humanAtrium bbs script