Introduction to the Workshop on Continuous Integration for High Performance Computing

By Edward Smith

10th November 2017

Plan for the Day

http://bit.ly/2zs189i

Session start	Title	Speaker
09:45:00	**Coffee and Registration**	
10:00:00	Introduction and plan for the day, Summary of Suggestions from sign-up form	Edward Smith
10:10:00	HPC and CI at Imperial	Spencer Sherwin
10:30:00	Challenges to Testing on HPC from my work on CPL library	Edward Smith
10:40:00	The SESC Build Service - CI for the UK computational science community	Steven Lamerton
11:00	Results of the STFC CI survey	Catherine Jones
11:20	Panel Guided Discussion -- what would be useful in a central service?	All above

Plan for the Day

http://bit.ly/2zs189i

11:50	**Lunch**	
12:50	Continuous Integration of Nektar++ with Buildbot	Chris Cantwell
13:10	HPC-CI infrastructure at Cambridge	Jeffrey Salmond
13:30	Brainstorming on the pros and cons of Travis, Buildbot, Jenkins, CircleCI, etc for HPC	All
14:00	**Catering -- Coffee, Tea and Biscuits**	
14:15	Continuous Integration for DL_POLY_4	Alin Elena
14:30	Overview of RSE and CI on HPC at UCL.	David Perez-Suarez
14:50	Break off with group discussion session on key problems	All
15:50	Discussion of outcome summary document and close	Edward Smith

Scope

- Continuous integration for scientific software
  - High performance code
  - Minimal automated or unit testing
  - Acceleration: MPI, OpenMP, CUDA, etc
- Deployment on HPC including:

  1) Best ways to automate building within the module environment
     - Use existing or build from source
     - Consider software and hardware changes

Scope

2) HPC specific problems for testing, which includes:

a)Initialisation through job submission scripts (Hook to github like Travis or a GUI interface like Jenkins)

b)Scaling (Unique HPC problem and efficiency bottlenecks to prevent wasted resources)

c)Jobs tested over varying numbers of processor (bytewise comparison needed/possible?)

d)Frequency of tests (github commit or user triggered?)

e)Location/queue for tests (Optimal use of resources is essential).

Aims/Outcome for Today

- An identification of the HPC and CI community.
- Discussion of best way to proceed by sharing experience
- This is split into

  1) For RSEs, how to implement this in the best way

  2) Getting users to start CI on HPC (e.g. can we make it as simple as using a Travis style .yml file)

- Produce a two page report to be shared on the SSI and RSE network on insights.
- Perhaps a move towards adoption of a similar approach across platforms and RSE groups.

Suggestions

- Project specific requirements vs national centralised facilities
- Best strategies for parallel and distributed software testing.
- Tutorials on CI frameworks (eg Travis)
- Development of a suitably flexible CI system managed within the College.
- CI deployments on the Cloud vs locally - Documentation of processes in different teams

Suggestions

- I'm interesting in attending in order to gather research support requirements that I can feed back to IC ICT
- I think talks should be limited and short. There is a lot to consider in this area, and it would be good to break up into "working groups" to discuss and present conclusions.
- Further suggestions:
  - http://bit.ly/2hg6NEM

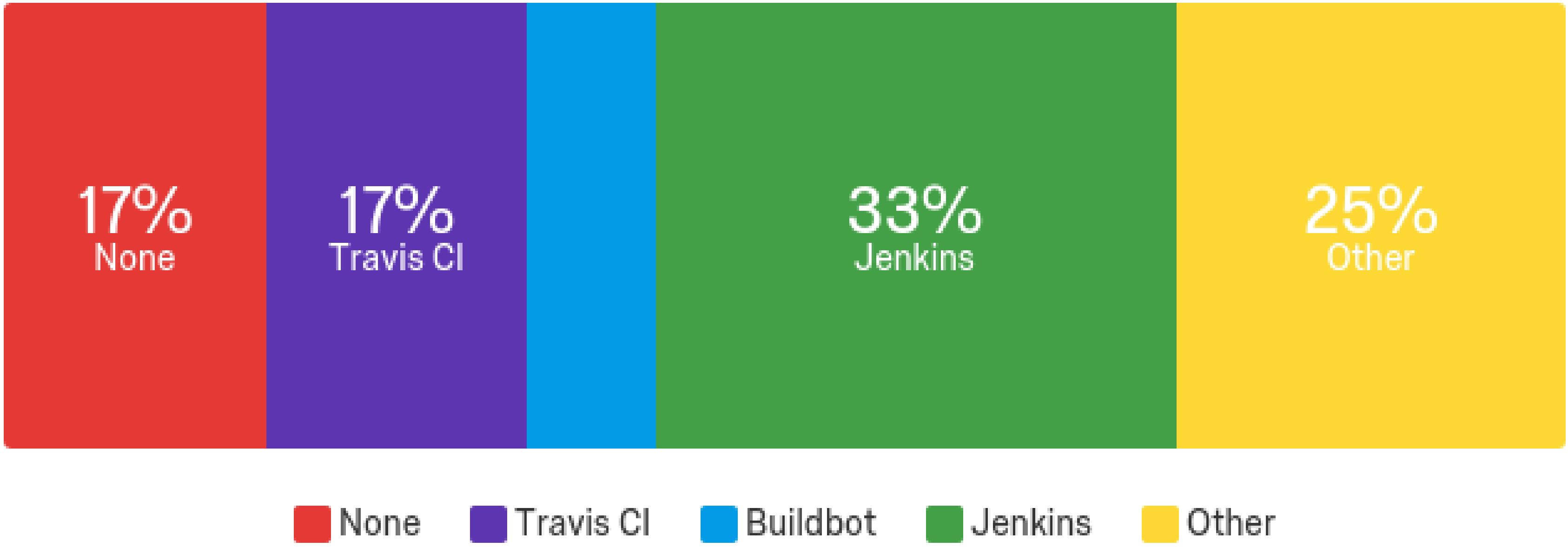
Work Group Discussions

- In the afternoon we have an hour for discussions
  - Please check the google document to see existing ideas
  - Add your own suggestions to the google docs or tweak what is there
  - Flexibility in how we use this time

Pros and Cons of CI frameworks

- In the afternoon we plan to brainstorm and identify the best CI for HPC
  - Pros and cons with emphesis on the particular challenges for HPC
  - Please think about this for the frameworks you've used, especially if you have worked with more than one
  - Would we consider writing our own?

Which Continous Integration frameworks



- Others include:
  - gitlab framework
  - kde framework (jenkins+customization)
  - Appveyor
  - Bamboo
  - circleci

Sponsors for Today

- Room supplied by

Imperial College London

- Funding for food and travel from

Software Sustainability Institute

Challenges to Testing on HPC from my work on CPL library

By Edward Smith

10th November 2017

Overview

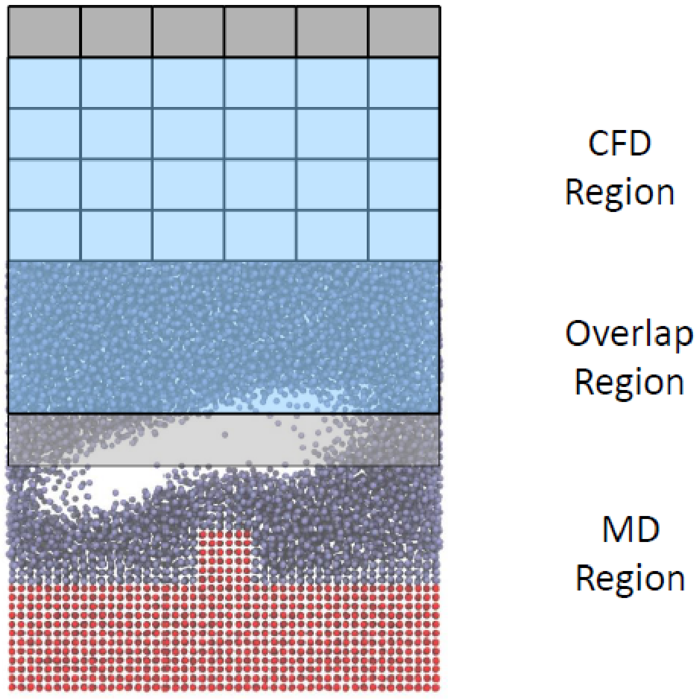
I will outline my experience, including:

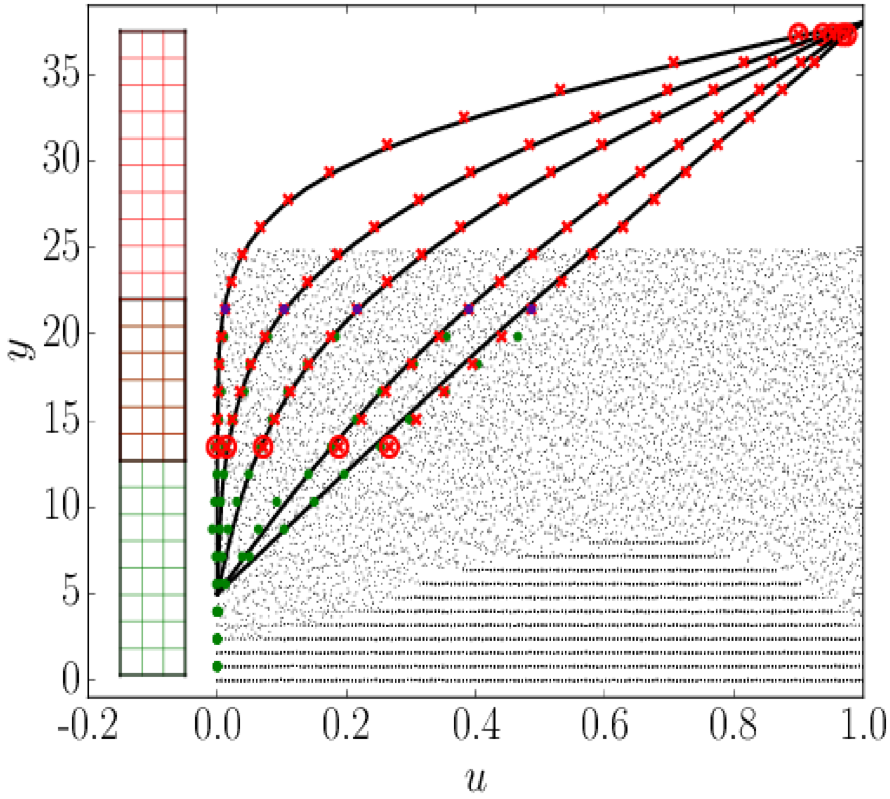
- Issues with building parallel codes
  - Cross platform Desktop, ARCHER, Imperial HPC and BP supercomputer in Texas
  - Cross language(C++, Fortran and Python)
  - Cross codebase - linking existing/evolving codes
- Issues with testing parallel code
  - Spawning parallel runs
  - Unit tests with parallel SetUp/TearDown
  - Ensuring parallel corner cases are tested
  - How to be sure test are meaningful?

CPL Library

- We are coupling two separate codes to run together
  - Computational Fluid Dynamics
  - Molecular Dynamics or Discrete Element Method
- Build codes separately and exchange all information as average fields through shared library (CPL library)
- This is good because it:
  - Allows separate testing of both codes
  - Maintains scope of both codes
  - Promotes optimal scaling

Domain Decomposition (MD near wall, CFD for remaining domain)





CPL LIBRARY

www.cpl-library.org

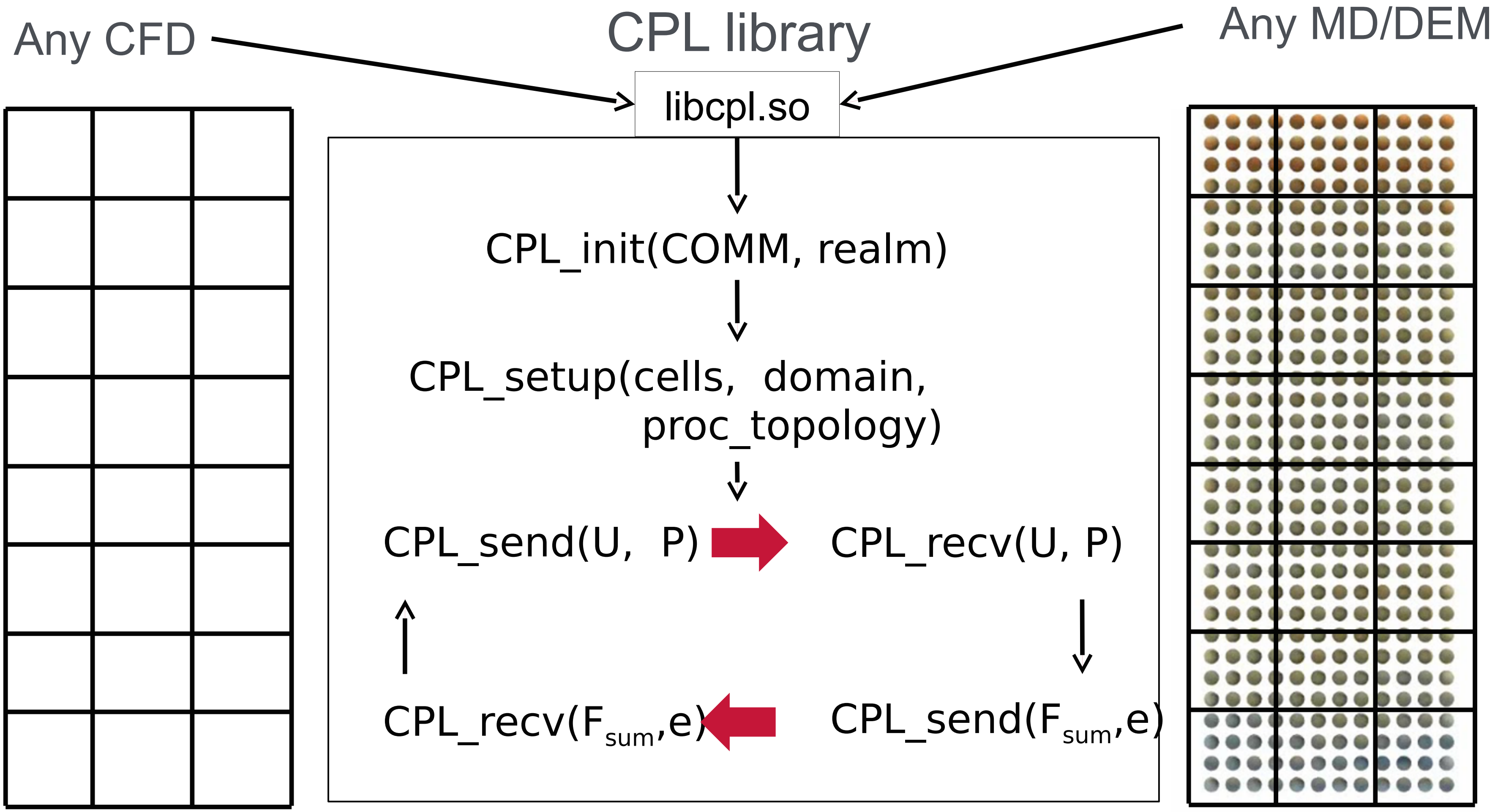
With David Trevelyan, Lucian Anton, Eduardo Ramos-Fernadez, David Heyes and Daniele Dini

Granular Mechanics and Computational Fluid Dynamics

- Discrete element method fully overlapping with CFD code with two way coupling through particle drag forces

With Catherine O'Sullivan

CPL Library - A Tale of Two Grids



Any CFD

CPL library

Any MD/DEM

libcpl.so

CPL_init(COMM, realm)

CPL_setup(cells, domain,
proc_topology)

CPL_send(U, P)

CPL_recv(U, P)

CPL_send(F_{sum},e)

CPL_recv(F_{sum},e)

Two codes sharing a communicator

mpiexec -n 4 ./cfd.exe : -n 48 ./dem.exe

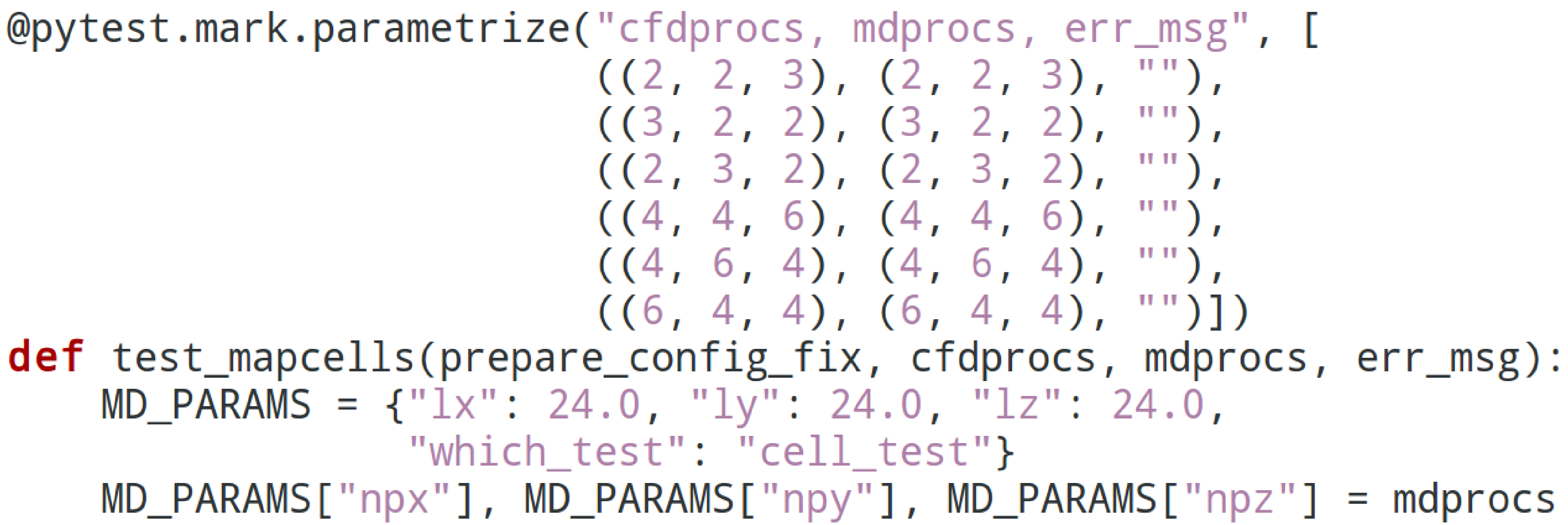
Unit testing CPL Library - Verification

Testing the basic units of code

- Tools to apply drag to particles, CFD boundary conditions, etc (All work in serial)
- Basic use of CPL_init, CPL_setup, CPL_send and CPL_recv
- Tested over a range of processor topologies

Unit testing CPL Library - Verification

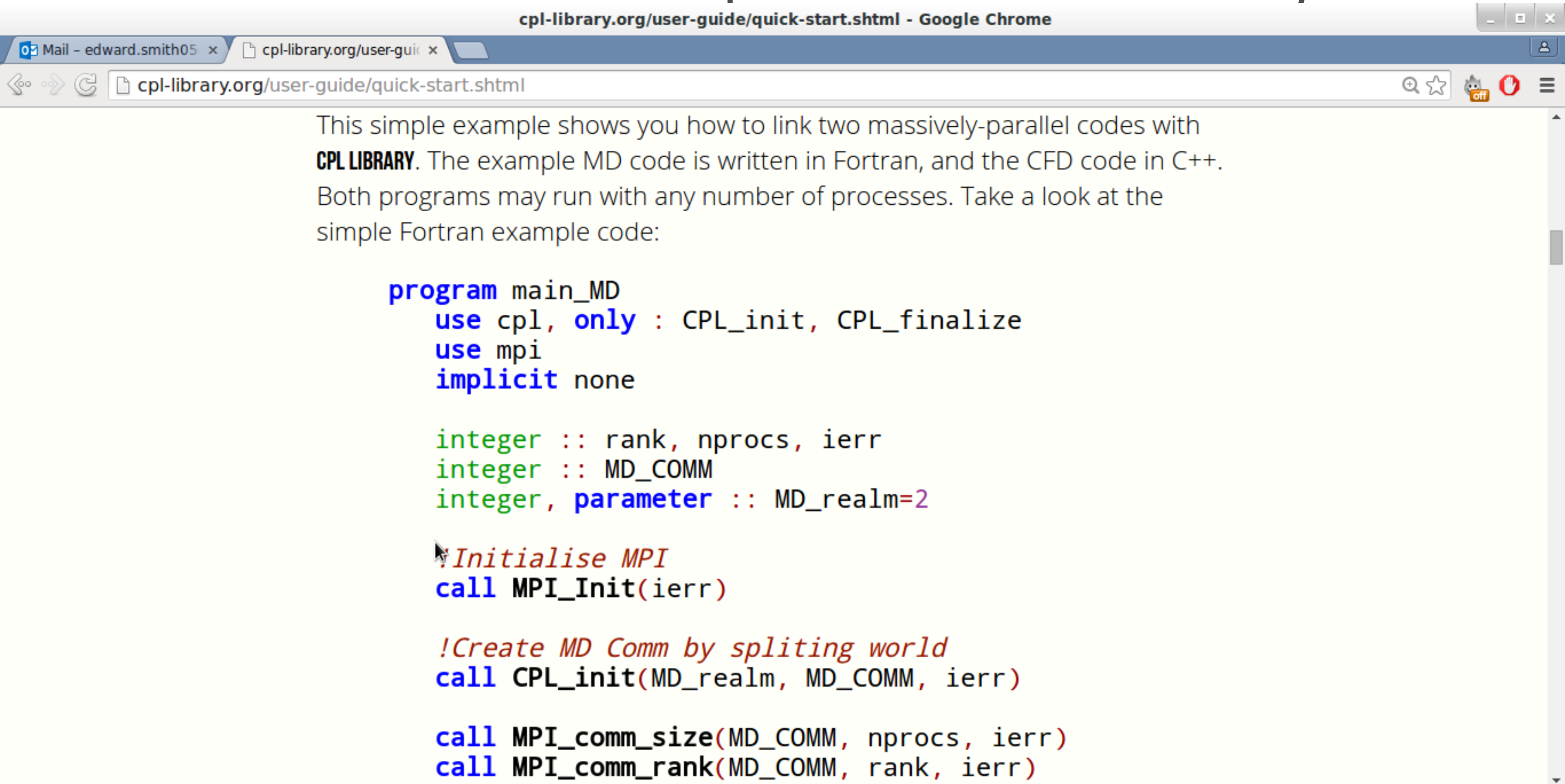
- We test parallel case on a range of different processor topologies
  - pytest parameterize to create all possibilities
  - Serial code uses subprocess (i.e. python starting another program) to create possible MPI runs with mpiexec



```
@pytest.mark.parametrize("cfdprocs, mdprocs, err_msg", [
                         ((2, 2, 3), (2, 2, 3), ""),
                         ((3, 2, 2), (3, 2, 2), ""),
                         ((2, 3, 2), (2, 3, 2), ""),
                         ((4, 4, 6), (4, 4, 6), ""),
                         ((4, 6, 4), (4, 6, 4), ""),
                         ((6, 4, 4), (6, 4, 4), "")])
def test_mapcells(prepare_config_fix, cfdprocs, mdprocs, err_msg):
    MD_PARAMS = {"lx": 24.0, "ly": 24.0, "lz": 24.0,
                 "which_test": "cell_test"}
    MD_PARAMS["npx"], MD_PARAMS["npy"], MD_PARAMS["npz"] = mdprocs
```

Testing the Examples

Examples are automatically stripped for website HTML and included as part of the CI testing


cpl-library.org/user-guide/quick-start.shtml - Google Chrome
Mail – edward.smith05
cpl-library.org/user-gui
cpl-library.org/user-guide/quick-start.shtml

This simple example shows you how to link two massively-parallel codes with CPL LIBRARY. The example MD code is written in Fortran, and the CFD code in C++. Both programs may run with any number of processes. Take a look at the simple Fortran example code:

```
program main_MD
   use cpl, only : CPL_init, CPL_finalize
   use mpi
   implicit none

   integer :: rank, nprocs, ierr
   integer :: MD_COMM
   integer, parameter :: MD_realm=2

   !Initialise MPI
   call MPI_Init(ierr)

   !Create MD Comm by spliting world
   call CPL_init(MD_realm, MD_COMM, ierr)

   call MPI_comm_size(MD_COMM, nprocs, ierr)
   call MPI_comm_rank(MD_COMM, rank, ierr)
```

Continuous Integrated Testing

- We use Travis CI as it runs automatically with github. If your latest change breaks the test, you will get an email
- A free service for open source projects
- Actually works quite well with MPI and parallel codes scaled up to 64 processors
- Based on Ubuntu 12.04 (newer libc than ARCHER!)
- Opaque technology with no way to test and develop locally – need to commit to trigger test

Continuous Integrated Testing

- Travis CI

```
# http://travis-ci.org/Crompulence/cpl-library
os: linux
sudo: required
language: python
python:
    - 2.7
env:
    - MPI=mpich3 GCC_VERSION=5
before_install:
    - sh ./make/travis/travis-install-gcc.sh
    ...
    - export MPI_DIR=$MPI_BUILD_DIR/$MPI
install:
    - export PATH=$MPI_DIR/bin:$PATH
    - make
script:
    - make test-all
```

Summary of Build Problems

- To run these tests on HPC
  - Shared library must be compiled with same version of MPI as two coupled codes
  - Differences in compiler for each linked code can cause problems
  - Loading module version is unreliable and we often have to patch the source code anyway (solution with MPI_port not supported on Cray)
  - Rebuilding both linked codes is often prohibitivly slow (e.g. OpenFOAM takes 8+ hours)
  - Problems getting Pytest setup on HPC

Deployment Through Anaconda

- A package manager used mainly for scientific computing and Python
- Coupled OpenFOAM /LAMMPS setup with a single script in minutes (vs. 8+ hrs to build from source)
- Packaged on virtual machine for compatibility with linux on all platforms: ARCHER, CX1, CX2 and BP's supercomputer in Texas
- Want to use these on HPC to for unit tests, scaling test and to check the physics is working correctly on varying topologies
- Container may be better?

This work is mainly by Eduardo Ramos-Fernandez

Summary of Test Problems

- Tesing multiple processors requires separate processor topologies to be spawned (we cannot start varying MPI jobs in a test framework)
- More complex under PBS, best way to do this?
- Scaling and efficiency are an important part of the CI testing, regression tests or against a benchmark
- Not always clear what should be tested: bytewise processor checks or regression? Physical laws? Input/Output on HPC systems (with efficiency)

A Workshop to Discuss CI Tesing on HPC

- Such complex integration of code on HPC will only be possible using modern programming practices
- If we don't automate testing, it doesn't get run!
- Some HPC Continous Integration in various places, e.g. Cambridge, MET office & starting on ARCHER.
- Questions:
  - How to address the unique problems of CI on HPC?
  - Can (or should) we make this as simple for users as setting up a .travis.yml file?
  - Better to help academics set this up themselves?