Progress on bringing High Performance Computing to MAP D&S

Robert Ryne

LBNL

Nov 2, 2012

Bringing High Performance Computing (HPC) to MAP D&S

- Incorporating HPC is an integral part of our D&S plans.
- Enables:
  - inclusion of more realistic physics models
  - running jobs with higher resolution
  - running jobs with higher statistics
  - running medium-scale jobs faster
  - scanning problem parameters (e.g. lattice parameters)
  - faster turn-around for human-driven optimization
  - computer-driven design optimization

Brute force computer power is not a replacement for being innovative. Goal is to use powerful computational tools AND be innovative in order to explore concepts and ultimately develop the best designs for MAP.

National Energy Research Scientific Computing Center (NERSC)

- About NERSC*:
  - the primary computing facility for the DOE Office of Science
  - provides HPC/information/data/communications services for research sponsored by DOE Office of Science
  - Largest system is Hopper, a Cray XE6 with 153,000 compute cores
- A new repository (called “map”) has been set up at NERSC
- A new project directory (/project/projectdirs/map/) enables sharing of data and codes, and a place to work in addition to $HOME and /scratch/
- We are now using Hopper (hopper.nersc.gov) to perform MAP simulations

*http://www.nersc.gov/

Obtaining a NERSC account for MAP modeling

First, email me:

- your name, institution, phone, preferred userid, citizenship

Next, go to

- http://www.nersc.gov/users/accounts/user-accounts/nersc-computer-use-policies-form/

Read and sign the form, then upload (preferred) or email or fax the form to NERSC

Accounts are usually set up within 24 hrs

NERSC "getting-started" notes

web pages, getting help:
NERSC has excellent web pages: http://www.nersc.gov
If you need to speak to a consultant, dial 1-800-66NERSC

To login:
```
ssh -Y hopper.nersc.gov -l youruserid
<enter your password>
```

To move files from a remote computer (such as your laptop or personal computer) using a Terminal window:

```
scp filename your_user_id@hopper.nersc.gov:.
<enter your password>
```

NERSC "getting-started" notes

To compile:

The fortran compiler is invoked via "ftn" and the C compiler is invoked using "cc"

See "Compiling Code" at www.nersc.gov/users/getting-started/

The default fortran compiler is PGI

Modules:

For detailed info see "Modules" in the middle of the page at http://www.nersc.gov/users/getting-started/

Here are the basics:

To see what is loaded, type

```
module list
```

To see what is available, type

```
module avail
```

NERSC "getting-started" notes

Submitting batch jobs:

```
qsub your_batch_script_file_name

Following the qsub command, you will
see the jobid that is assigned to it
```

Monitoring jobs:

```
qs -u your_user_id
```

To kill an already submitted job:

```
qdel your_job_id
```

Sample batch script:

```
#PBS -q debug
#PBS -A map
#PBS -l mppwidth=512
#PBS -l
walltime=00:30:00
cd $PBS_O_WORKDIR
aprun -n 512 ./icool.x
```

Interactive jobs:

www.nersc.gov/users/computational-systems/hopper/running-jobs/interactive-jobs/

MAP project directory

- /project/projectdirs/map/
- map/

  Archives/ Baseline/ Benchmarks/ Codes/ Geant4Data Users/
- Baseline/

  ProtonDriver/ FrontEnd/ Cooling/ AccStorage/ Collider/
- Codes/

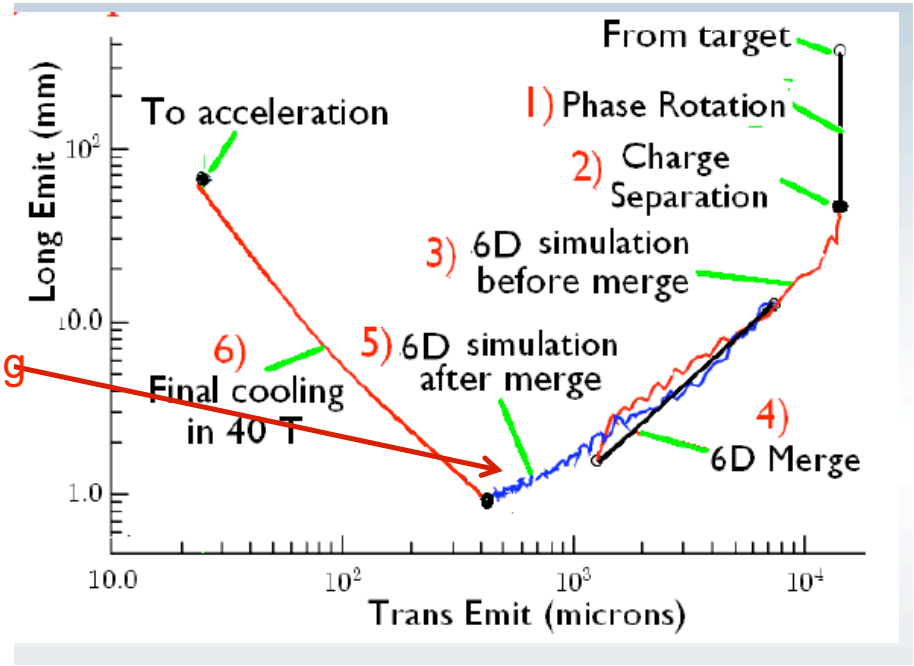
  icool330/ icool331a icool_warp/ ScanParams/ Warp_input/ …
- Users/

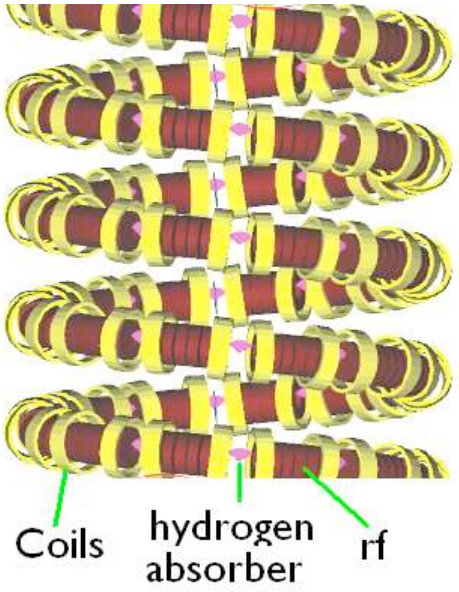
  diktys/ hsayed/ ryne/ tjrob/ …

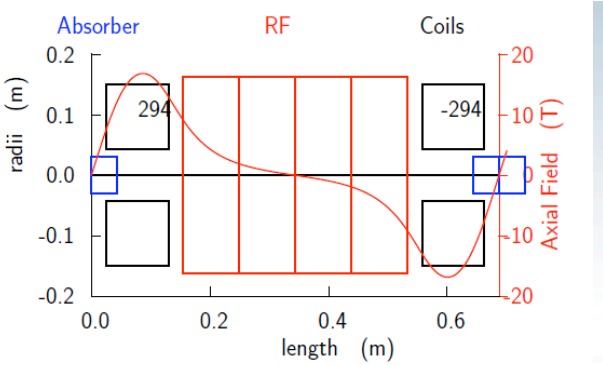
Accomplishments

- Serial version of ICOOL 3.31 installed
- Parallel version of G4BL installed
  - major performance improvements achieved (see Tom Roberts talk)
- Developed scripts to perform parallel parameter scans
- Archived previous NF and MC designs
- Extracted ICOOL beam-material interaction (BMI) routines
- Incorporated BMI routines in Warp
- ICOOL Front-End simulations underway
- Warp cooling simulations w/ space charge underway

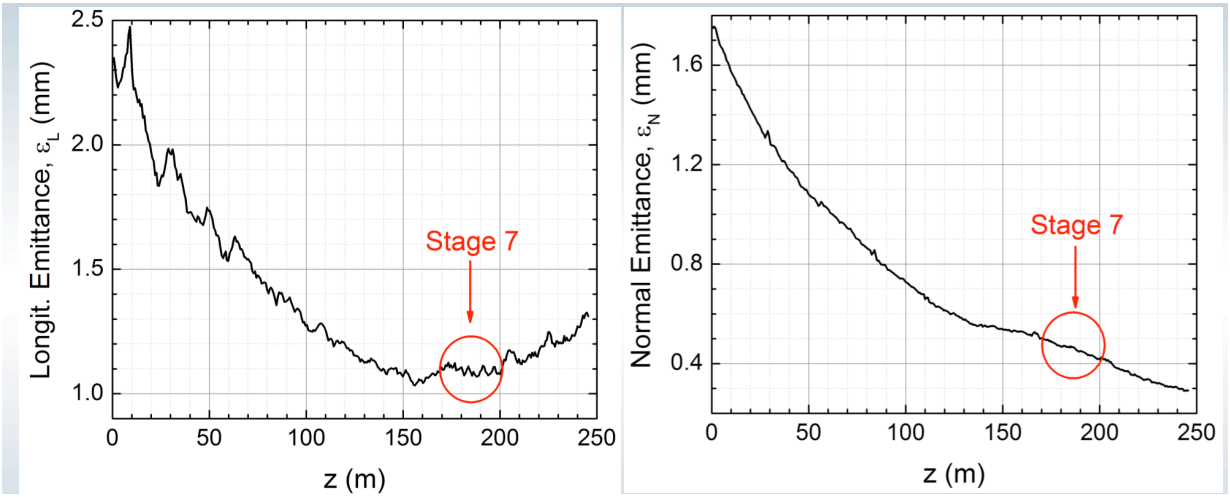
Simulation of 6D cooling with space charge (D. Stratakis, D. Grote)





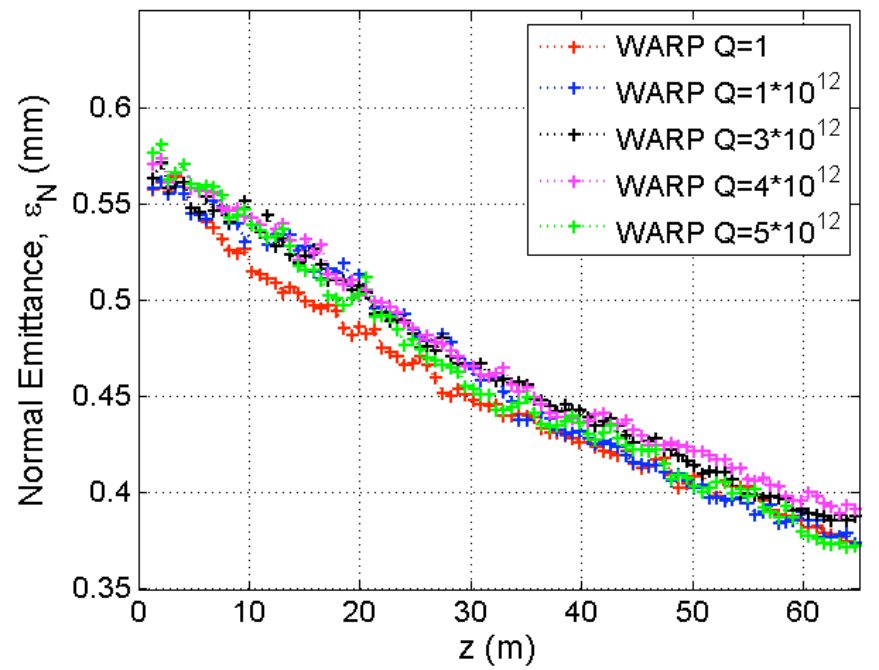


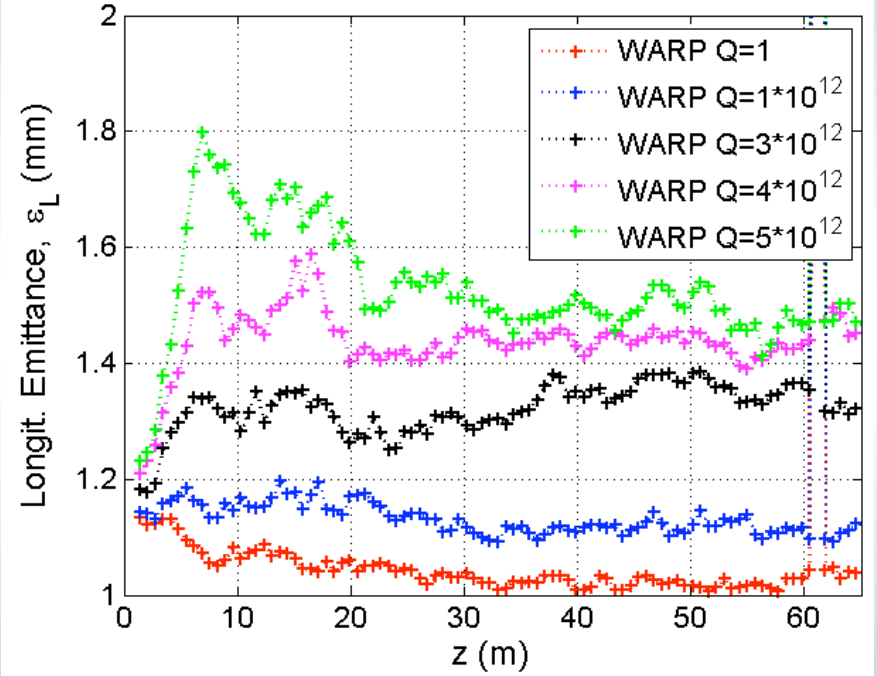
ICOOL sim of 8 stages of cooling w/ a field-flip lattice. Each stage contains ~50 cells



Initial studies look at Stage 7, use Warp to examine effects of space charge

Muon Cooling with Space-Charge (1)





D. Stratakis, D. Grote

Tutorial on running parallel parameter scans

- Very little extra work beyond what you would do anyway to run a single job
  - 1. prepare your input files as usual
  - 2. prepare 1 new (small) input file describing what parameters are to be varied
  - 3a. make minor edits to a batch script (example provided) to run multiple serial jobs

  or

  - 3b. make minor edits to a unix script (example provided) to run multiple parallel jobs

Parameter scans with a serial executable

1. Login to Hopper
2. Prepare your input files as usual
3. Create a new file called "scanparams.in" describing parameters to be scanned, plus names of any other input files
4. Create a PBS batch script. Here it is called `scanscript`
   - edit `scanscript` to set walltime, queue, exec code
5. Submit `scanscript`

Results will appear in separate subdirectories. A list of the parameters used for each case will be found in a file called "scanparams.out"

Note: To copy a sample version of `scanscript` :

```
cp /project/projectdirs/map/Codes/Scanparams/scanscript .
```

Parameter scans with a parallel executable

1. Login to Hopper
2. Prepare your input files as usual
3. Create a new file called "scanparams.in" describing parameters to be scanned, plus names of any other input files
4. Create a unix script. Here it is called `pscanparams`
   - edit `pscanparams` to set walltime, queue, exec code, cores/exec
5. Run `pscanparams`
   - `this will create a batch script and submit it for you`

Only difference compared to the serial case is that you
(1) edit a unix script instead of a PBS batch script, and
(2) also specify the # of cores per parallel executable

Results will appear in separate subdirectories. A list of the parameters used for each case will befound in a file called "scanparams.out"

Note: To copy a sample version of `pscanparams` :

```
cp /project/projectdirs/map/Codes/Scanparams/pscanparams .
```

Example with a serial exectuable: scanning two quantities in an ICOOL run

- Login to Hopper: `ssh –Y hopper.nersc.gov –l your_user_name`
- Prepare your input files as usual
- Create a new file called "scanparams.in" describing parameters to be scanned, plus names of any other input files*

```
for001.dat float 0.30 0.40 24 's/0.365 1 .001/############ 1 .001/'
for001.dat float 0.01  0.03  20 's/0.02 1 .001/############ 1 .001/'
```

*note: to specify a file for which no parameters are varied, just list the name followed by / `ptcls.in /`

- Edit PBS batch script, called `scanscript`, to set execution time, queue, and location of serial executable

```
#PBS -q debug
#PBS -A map
#PBS -l mppwidth=480
#PBS -l walltime=00:03:00
cd $PBS_O_WORKDIR
aprun -n 480 /project/projectdirs/map/Codes/Scanparams/scanparams.x /project/projectdirs/map/Codes/icool330/icool
```

you edit this

you edit this to point to your serial executable

edit to match above

- Submit batch script: `qsub scanscript`

Example with a serial exectuable: scanning the seed in an ICOOL run

- File "scanparams.in" could look like this:

```
for001.dat integer -1  -10000  480 's/rnseed=-1/rnseed=#########/'
```

- Note: In general, besides scanning using `real` and `integer` you can also specify `octal`

Example with a parallel exectuable: scanning two quantities in an MLI run

- Same as for the serial case, you create a file "scanparams.in"

```
mli.in float 0.30d0 0.40d0 4 's/dr365: drift, l=0.365  slices=36/dr365: drift, l=###############  slices=36/'
mli.in float 0.01d0  0.03d0 5 's/dr02: drift, l=0.02  slices=20/dr02: drift, l=###############  slices=20/'
```

- Instead of editing the PBS script called `scanscript`, you edit the beginning and end of a unix script called `pscanparams`

```
#!/bin/bash -l

nprocs_per_job=24
walltime='00:04:00'
queue='premium'
…
…
…
…
echo "aprun -n $nprocs_per_job ~/MLIjuly2012/mli.x >& std$m &" >> qscript
…
```

you edit this

you edit this to point to your parallel executable

- Launch the job (i.e. run the unix script) by typing: ./pscanparams