SC20

Everywhere we are | more than hpc.

Practical Persistent Memory Programming

Adrian Jackson
EPCC
a.jackson@epcc.ed.ac.uk
@adrianjhpc

epcc

Rosa Badia
BSC

Javier Conejero
BSC
francisco.conejero@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Aims

- Understand persistent memory hardware and software
- Understand the different aspects that can impact performance, and the shared/private nature of the resources
- Learn how to program persistent memory
- Get hands on with persistent memory hardware

Aims cont

- Understand data movement and think about application data requirements
- Understanding I/O and data movement, particularly persistence requirements, is hard
- Thinking about different ways you can do I/O or storing data

Format

- Lectures and practicals
- Slides and exercise material available online:
  - https://github.com/NGIOproject/PMTutorial
  - Exercises will be done on remote machine (NEXTGenIO prototype)
  - You should have accounts for these, or contact us live if you don't

Timetable

- 10.00 Introduction
- 10.15 Practical: IOR and Streams
- 10.30 Hardware, I/O and storage
- 11.00 Practical: Using different mount points
- 11.15 Low-level persistent memory programming
- 11.30 Break
- 12.00 Practical: Persistent memory programming
- 12.30 Persistent thinking
- 12.45 Higher-level persistent memory programming
- 13.15 Persistent Memory Programming with PyCOMPSs
- 14.00 Summary and finish

Programming persistent memory

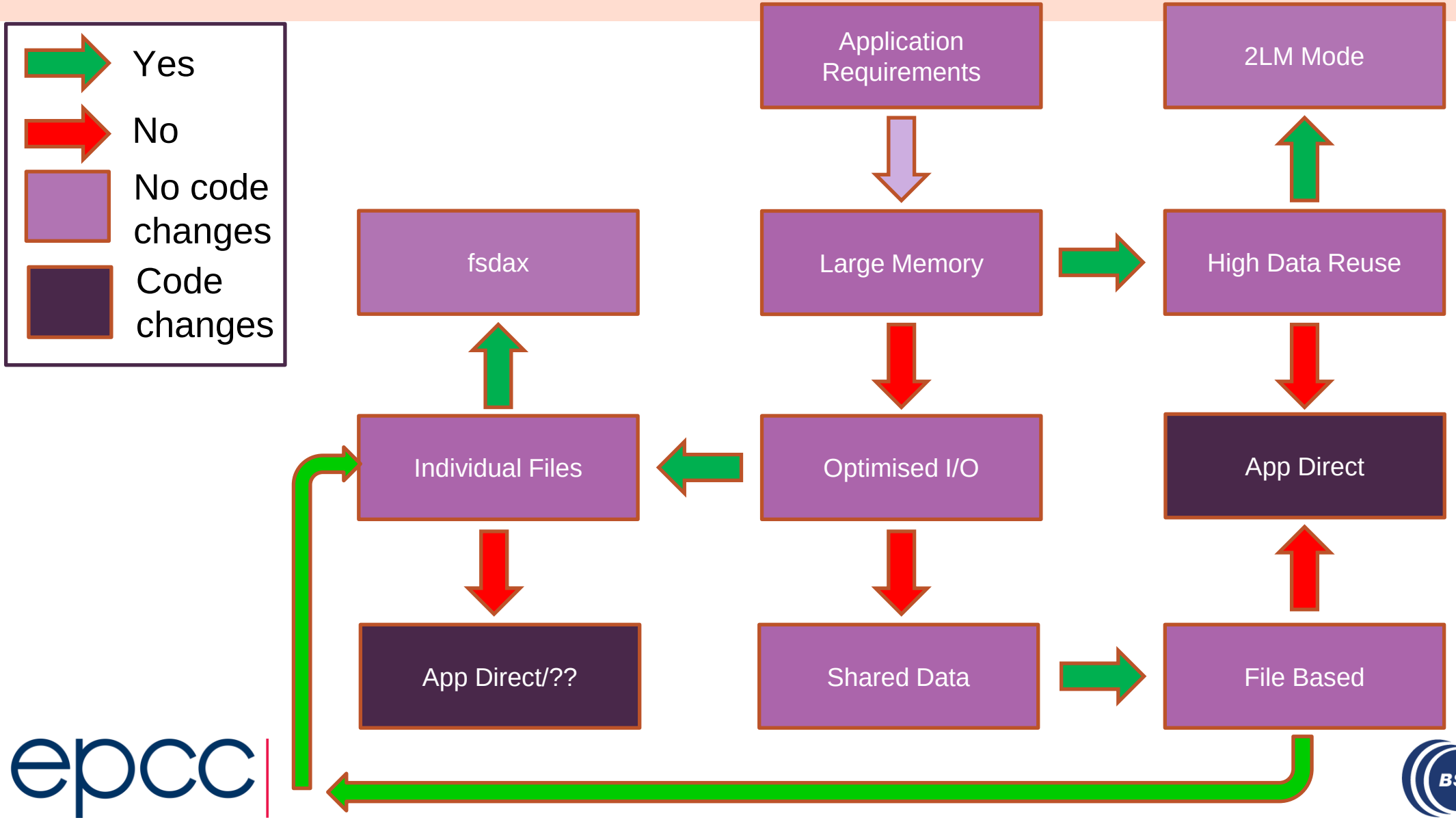
Programming persistent memory

```
double *a, *b, *c;
pmemaddr = pmem_map_file(path, array_length,
                         PMEM_FILE_CREATE|PMEM_FILE_EXCL,
                         0666, &mapped_len, &is_pmem)
a = pmemaddr;
b = pmemaddr + (*array_size+OFFSET)*BytesPerWord;
c = pmemaddr + (*array_size+OFFSET)*BytesPerWord*2;

#pragma omp parallel for
for (j=0; j<*array_size; j++){
   a[j] = b[j]+scalar*c[j];
}
pmem_persist(a, *array_size*BytesPerWord);
```

Programming persistent memory

Programming Persistent Memory



Programming persistent memory

- Design and performance considerations are the challenge
  - Programming the memory is easy
- Design for functionality
  - What is persistent, when is it persistent, what failures can you tolerate, etc..
- Design for performance
  - Memory size, I/O, data access costs, etc…
- Design for hardware configurations
  - NUMA, filesystems, storage, etc…

Summary

- Please don’t hesitate to ask questions!

- We are doing practicals
  - But if you’re not confident in programming we have other options

Practical

- Take IOR and STREAMS source code
- Run on the prototype
- Prototype is available at:

```
ssh hydra-vpn.epcc.ed.ac.uk
```

- Then

```
ssh nextgenio-login2
```

- Using your guest account
- Practical source code is available at:

```
/home/nx01/shared/pmtutorial/exercises
```

- Using slurm as the batch system
  - `pm` partition for this course

```
sbatch --reservation pm scriptname.sh
```