TUTORIAL

High Performance Object Storage: I/O for the Exascale Era

Johann Lombardi
Mohamad Chaarawi

Adrian Jackson
(a.jackson@epcc.ed.ac.uk)

Aims

- Understand storage hardware and software
- Learn how to program, and design programs, for storage systems
- Learn about DAOS and Ceph
- Learn about DAOS and Ceph apis and exploiting them from your applications
- Get hands on with DAOS, Ceph, and high performance storage hardware

Aims cont.

- Understand data movement and think about application data requirements
- Thinking about different ways you undertake I/O or storing data
- Move beyond bulk, block-based, I/O paradigms

Format

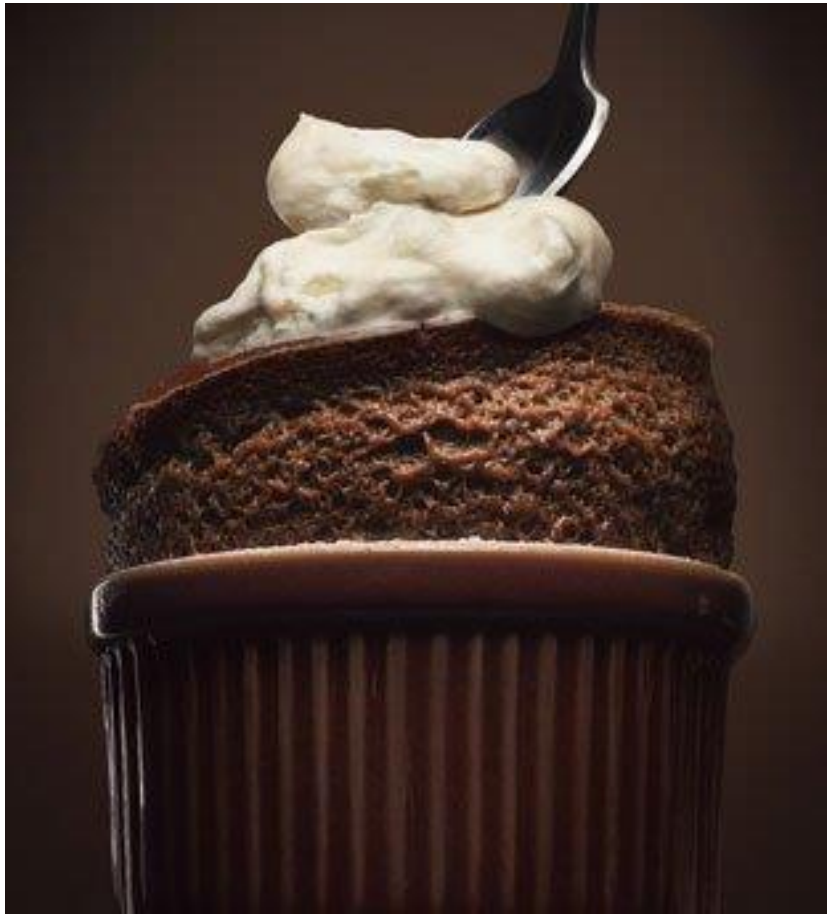
- Lectures and practicals
- Slides and exercise material available online:
  - https://github.com/NGIOproject/ObjectStoreTutorial
  - Exercises will be done on remote system (GCP)
  - We will provide accounts for these

intel ECMWF epcc

Timetable

- 08.30 Introduction
- 08.45 Storage hardware and software
- 09.00 Practical: Benchmarking different storage approaches
- 09.30 Overview of object stores, DAOS, and Ceph
- 10.00 Break
- 10.30 DAOS API and programming object stores
- 11.00 Ceph storage interfaces and librados
- 11.30 Exploiting DAOS and/or Ceph for applications
- 12.00 Summary and finish

Object Storage



intel

ECMWF

epcc

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);
```

Using DAOS

```
mkdir /tmp/my_filesystem
dfuse -m /tmp/my_filesystem --pool tutorial --cont adrians

…

fusermount3 -u /tmp/my_filesystem
```

Programming DAOS

```
rc = daos_init();
daos_pool_info_t pool_info;
daos_cont_info_t co_info;
rc = daos_pool_connect(o->pool, o->group, DAOS_PC_RW, &poh,
&pool_info, NULL);
rc = daos_cont_open(poh, o->cont, DAOS_COO_RW, &coh, &co_info, NULL);
rc = dfs_mount(poh, coh, O_RDWR, &dfs);
rc = dfs_write(dfs, obj, &sgl, off, NULL);
rc = dfs_read(dfs, obj, &sgl, off, &ret, NULL);
rc = dfs_umount(dfs);
rc = daos_cont_close(coh, NULL);
rc = daos_cont_destroy(poh, o->cont, 1, NULL);
rc = daos_pool_disconnect(poh, NULL);
rc = daos_fini();
```

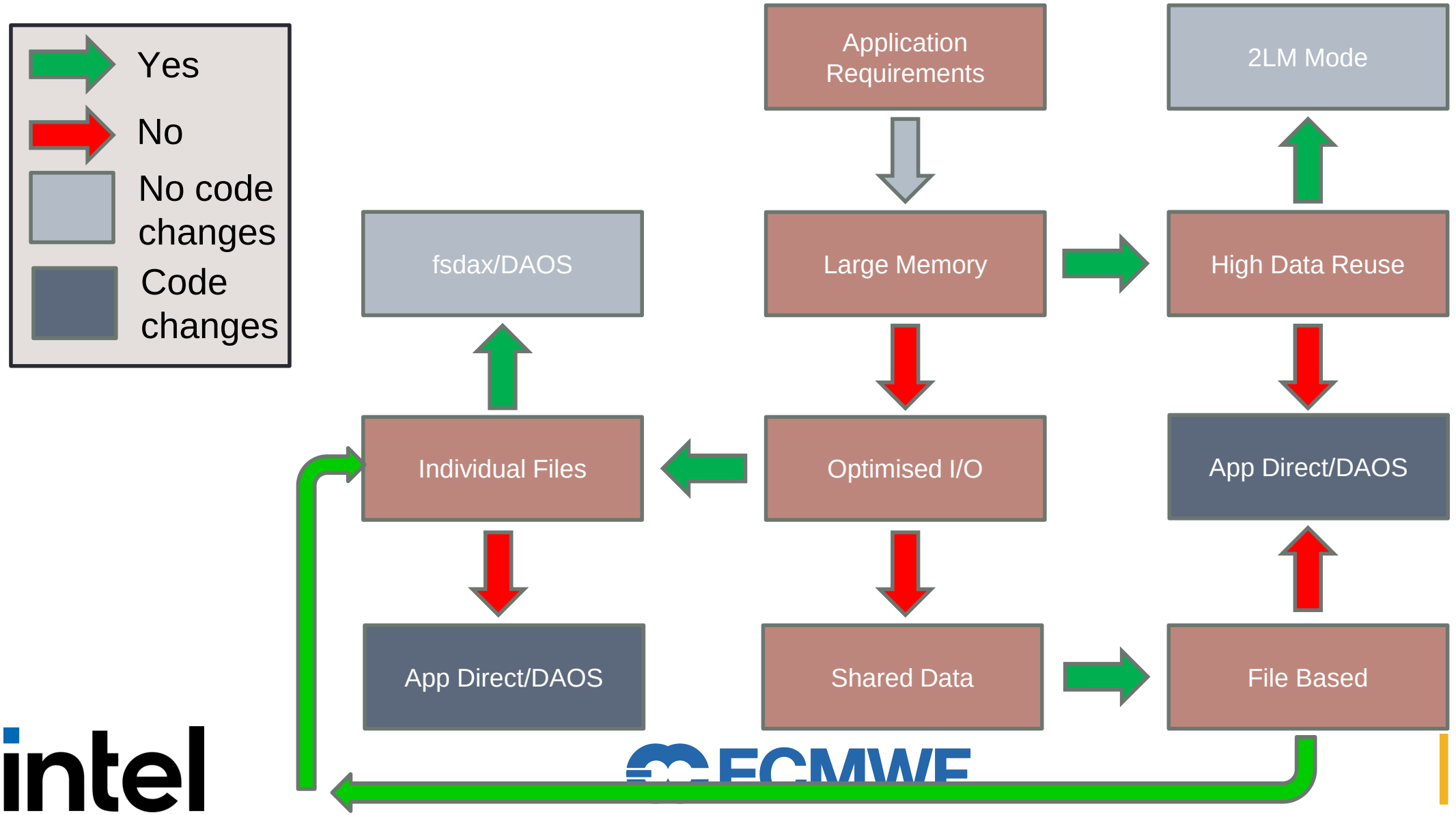
Practical Object Storage

intel

ECMWF

epcc

Programming Persistent Memory



intel

ECMWF

epcc

Object Storage

- Design and performance considerations are the challenge
  - Programming against the interfaces is relatively easy
- Design for functionality
  - What is persistent, when transactions should happen, what granularity I/O operations should be, 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...

Object storage

- Data stored in unstructured objects
  - Data has identifier
  - Size and shape can vary
  - Metadata can also vary
- Originally designed for unstructured data sets
  - Bunch of data with no specific hierarchy required
- Can also enable efficient/fast access to data in different structures
  - Supports different creation, querying, analysis, and use patterns

Summary

- Please don’t hesitate to ask questions!
- We have a practical sessions
  - Login account will remain active for you to try out using Ceph and DAOS after the tutorial
  - Email a.jackson@epcc.ed.ac.uk to get an account on the system we will use for practical/try out sessions