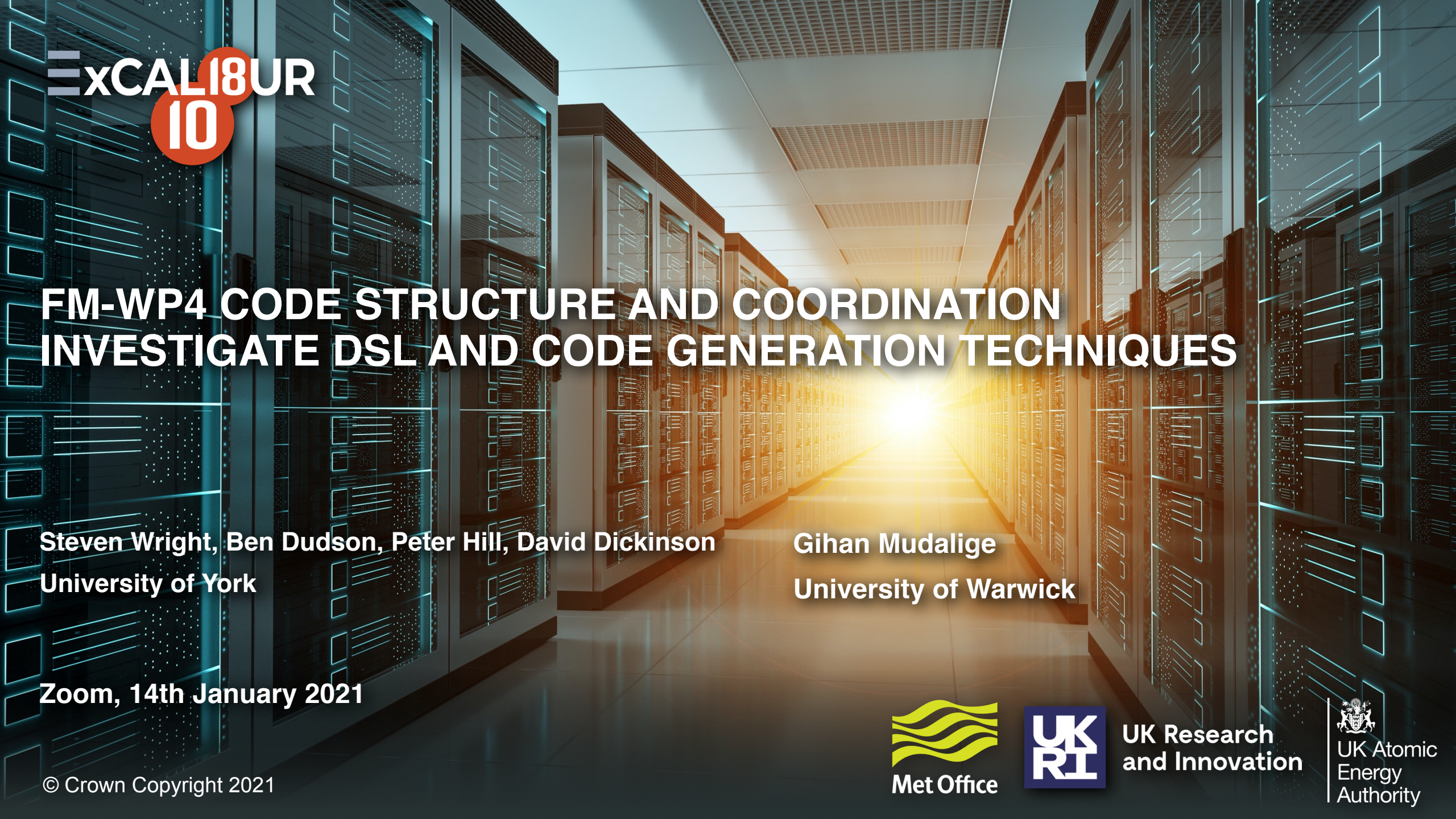


# FM-WP4 CODE STRUCTURE AND COORDINATION INVESTIGATE DSL AND CODE GENERATION TECHNIQUES

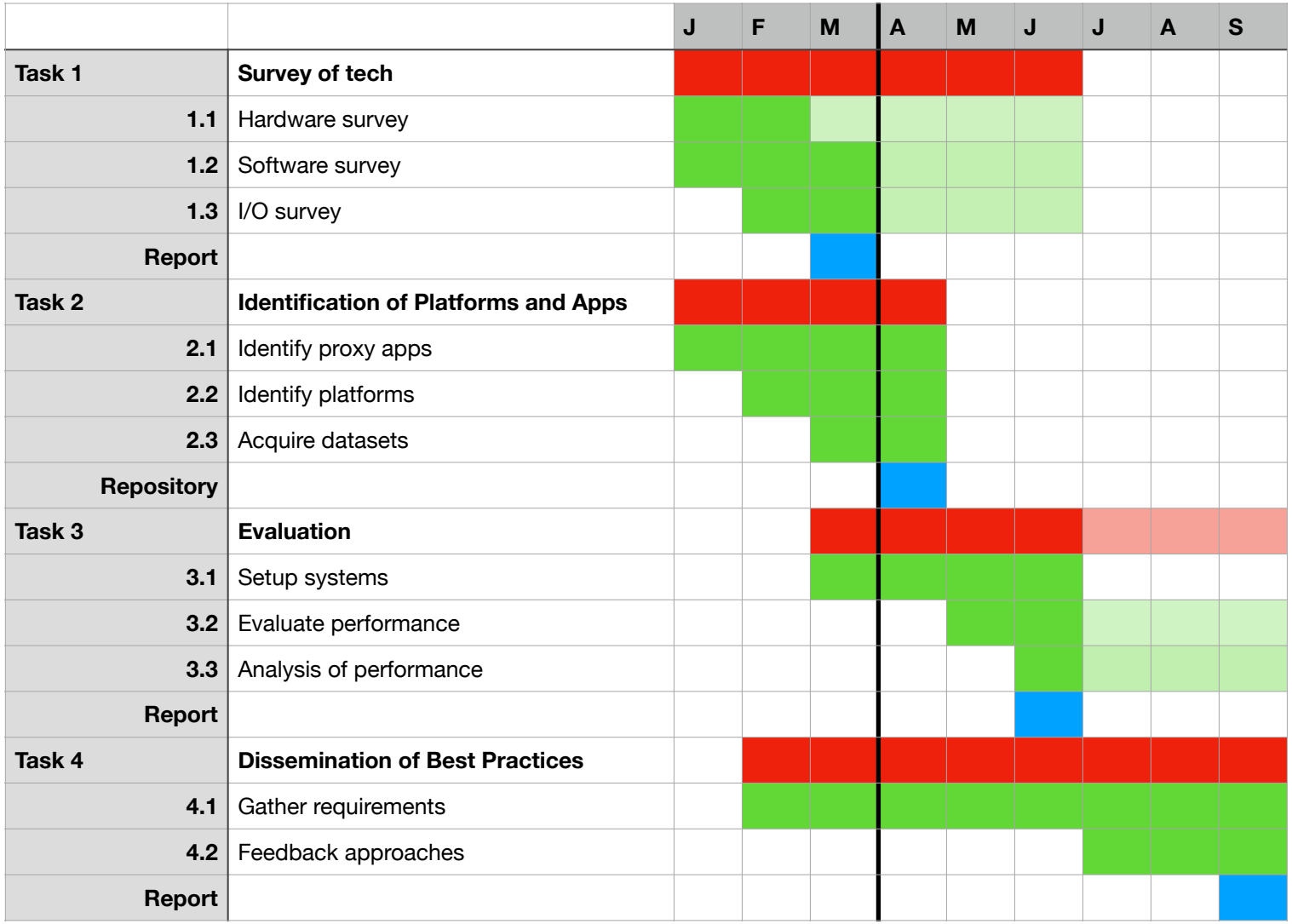

Steven Wright, Ben Dudson, Peter Hill, David Dickinson
University of York

Gihan Mudalige
University of Warwick


Zoom, 14th January 2021

© Crown Copyright 2021

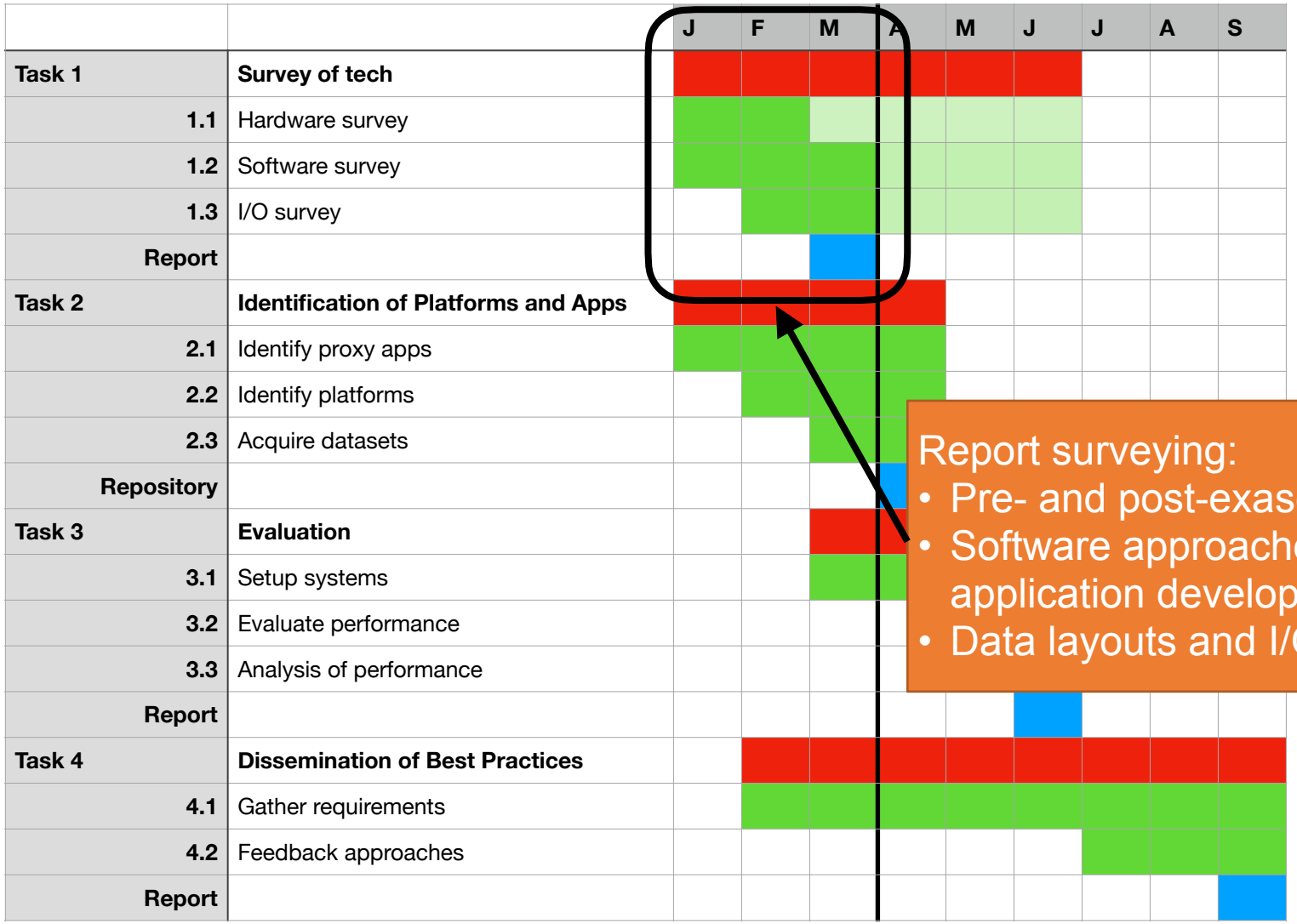
# Project Timeline




© Crown Copyright 2021

# Task 1

## Survey of available technology



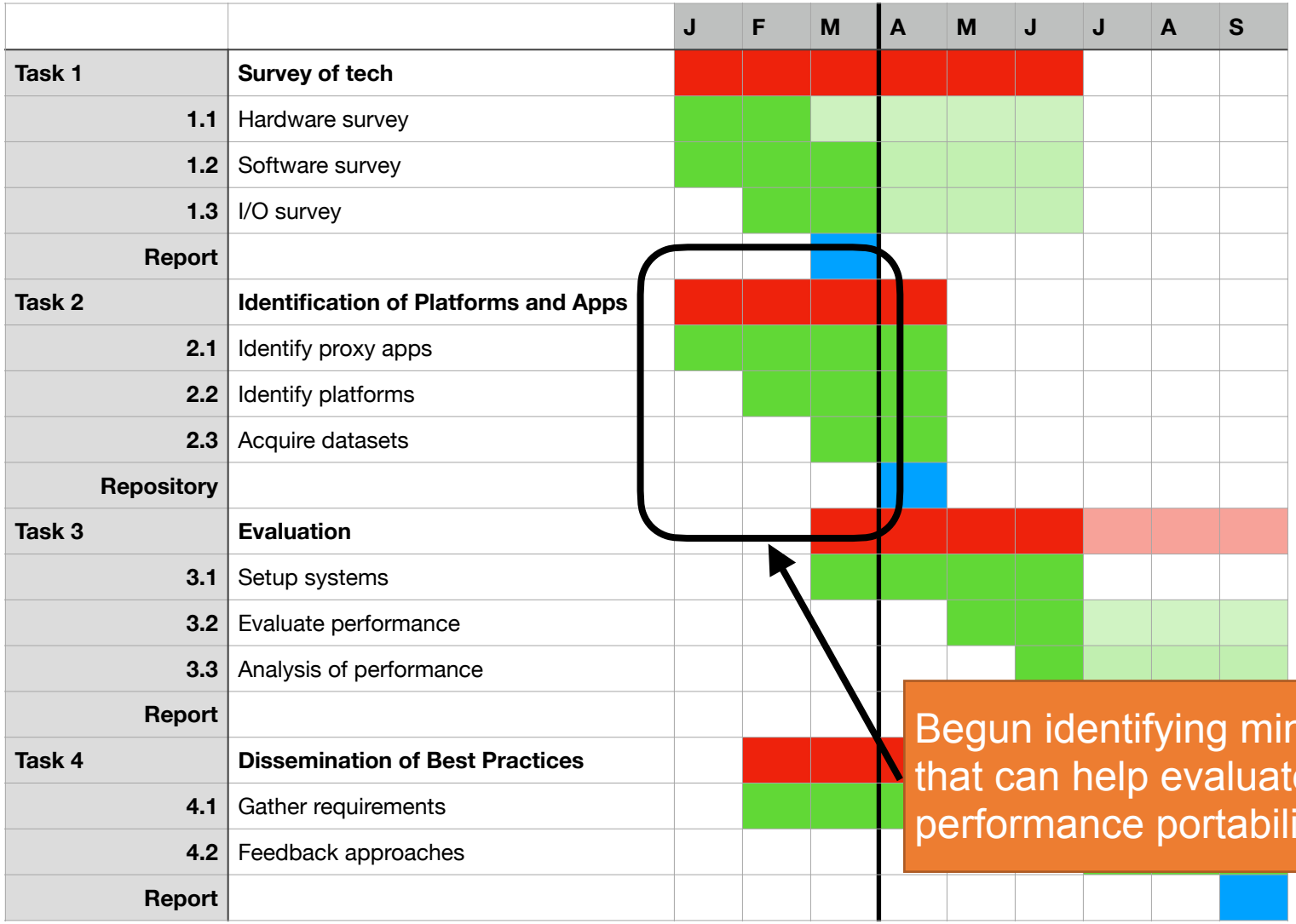

Report surveying:

- Pre- and post-exascale hardware
- Software approaches to Exascale application development
- Data layouts and I/O

© Crown Copyright 2021

# Task 2

## Identification of Platforms and Applications




Begun identifying mini-apps/proxy-apps that can help evaluate approaches to performance portability.

© Crown Copyright 2021

# Task 1.1

## Pre- and Post-Exascale Hardware

- All pre- and post-Exascale systems will be heterogenous (Except Fugaku)
- Most of the FLOP/s will be provided by GPU accelerators
  - NVIDIA (Lovelace/Hopper/Einstein?)
  - AMD Instinct
  - Intel Xe
- Most systems will use x86_64 hosts from Intel and AMD


© Crown Copyright 2021

# Task 1.1

## Pre- and Post-Exascale Hardware


DoE have 4 Cray Shasta systems in the pipeline

- All Cray Shasta Systems with Slingshot Fabric
- Perlmutter
  - >100 PFLOP/s
  - AMD EPYC CPUs with NVIDIA A100 GPUs
- Aurora
  - 1 EFLOP
  - Intel Xeon CPUs (Sapphire Rapids) with Intel Xe GPUs
- Frontier
  - 1.5 EFLOP/s
  - AMD EPYC CPUs (Milan) with AMD Instinct GPUs
- El Capitan
  - 2 EFLOP/s
  - AMD EPYC CPUs (Genoa) with AMD Instinct GPUs

© Crown Copyright 2021

# Task 1.1

## Pre- and Post-Exascale Hardware

**Europe has 3 pre-Exascale systems in development**

- **LUMI**
  - **0.5 EFLOP/s**
  - **AMD EPYC CPUs with AMD Instinct GPUs**
- **LEONARDO**
  - **0.3 EFLOP/s**
  - **Intel Xeon CPUs (Sapphire Rapids) with NVIDIA A100 GPUs**
- **MareNostrum 5**
  - **2 distinct 100+ PFLOP/s systems**
  - **Unknown architectures, but possible use of ARM/RISC-V**

© Crown Copyright 2021

# Task 1.1

## Pre- and Post-Exascale Hardware

In the UK,

- ARCHER-2
  - ~28 PFLOP/s
  - AMD EPYC CPUs
  - Cray Slingshot Fabric
- Many Tier-2 systems
  - Isambard-2
    - A64FX, AMD EPYC, NVIDIA GPUs, AMD GPUs, etc
  - Bede, Viking, Avon, etc

© Crown Copyright 2021

# Task 1.1

## Software Approaches to Exascale

- Hardware is diversifying rapidly…
- Portable software more important than ever before!
- Many approaches,
  - OpenMP/OpenACC directive-based methods
  - OneAPI/SYCL/OpenCL programming models
  - Kokkos/RAJA-style template libraries
  - PETSc/BLAS-style kernel libraries
  - OP2/OPS-style domain specific languages


© Crown Copyright 2021

# Task 1.1

## Software Approaches to Exascale

- Some Domain Specific Languages are low-level APIs aimed at developers
  - e.g. OP2 is a programming abstraction for unstructured mesh algorithms
- Some are high-level, aimed at allowing scientists to express the equations they're trying to solve
  - e.g. UFL, BOUT++

```
//main time-marching loop
for(int iter=1; iter<=niter; iter++) {
    // save old flow solution
    op_par_loop(save_soln, ... );

    // predictor/corrector update loop
    for(int k=0; k<2; k++) {

      // calculate area/timstep
      op_par_loop(adt_calc, ... );

      // calculate flux residual
      op_par_loop(res_calc, ... );
```

$$\frac{\partial \rho}{\partial t} = -\mathbf{v} \cdot \nabla \rho - \rho \nabla \cdot \mathbf{v}$$
$$\frac{\partial p}{\partial t} = -\mathbf{v} \cdot \nabla p - \gamma p \nabla \cdot \mathbf{v}$$
$$\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v} \cdot \nabla \mathbf{v} + \frac{1}{\rho}(-\nabla p + (\nabla \times \mathbf{B}) \times \mathbf{B})$$
$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

can be written simply as:

```
ddt(rho) = -V_dot_Grad(v, rho) - rho*Div(v);
ddt(p)   = -V_dot_Grad(v, p) - g*p*Div(v);
ddt(v)   = -V_dot_Grad(v, v) + (cross(Curl(B),B) - Grad(p))/rho;
ddt(B)   = Curl(cross(v,B));
```

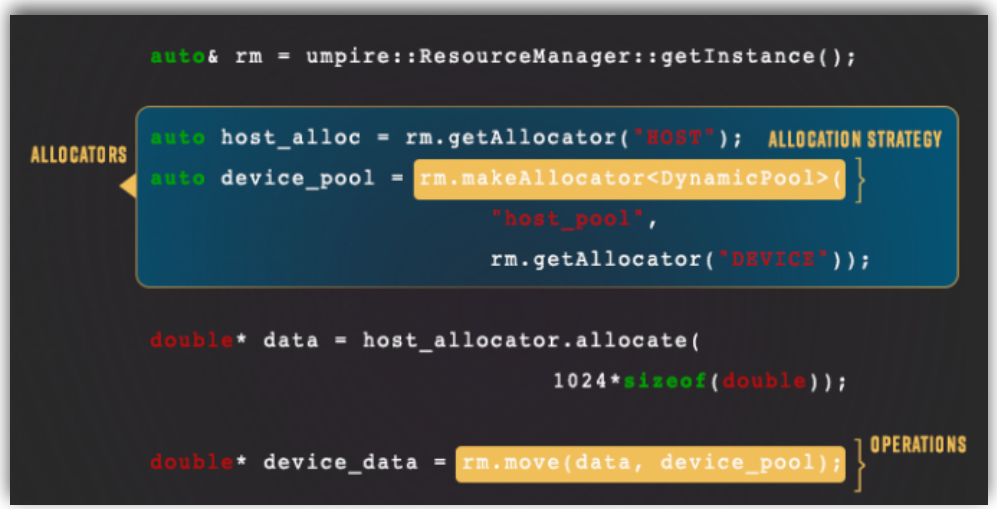
© Crown Copyright 2021

# Task 1.1

## Data representation and I/O

```
// A Default 3D-View with 2 runtime dimensions.
// This View will be reference counted
Kokkos::View<double**[8]> A(?A-Default?,1000,256);

// An atomic view of A. Every access (=,+=,=-,?) will be atomic.
// This View will be reference counted (i.e. it has a shared reference count with A
Kokkos::View<double**[8], Kokkos::MemoryTraits<Kokkos::Atomic> A-Atomic = A;
```




- With hierarchical parallelism, optimised data representation and movement are vital
  - Managing data transparently to the user
    - Kokkos Views
    - UMPIRE
    - Intel SDLT
  - API access that can be optimised for host architectures to ensure coalesced or strided accesses (where appropriate)

© Crown Copyright 2021

# Task 1.1

## Data representation and I/O

- I/O improvements typically lag compute improvements
- Standardised output format libraries
  - HDF5, netCDF
- Libraries for accelerated I/O
  - TyphonIO, SILO, ADIOS

- New approaches are emerging: e.g. NVMe, VeloC, Rabbit

© Crown Copyright 2021

# Task 1.1

## Conclusions

- Rewriting legacy codes for new heterogenous architectures 👎
- NEPTUNE is an opportunity to prepare a future-proofed code
- Important that we:
  - choose a programming model or DSL that supports many (or all!) backends 👍
  - choose an approach to data management that is hardware agnostic 👍
  - choose a user-interface (or DSL) that allows scientists to express problems without being overly restrictive 👍

© Crown Copyright 2021

# Task 2

## Identification of testbed platforms and representative proxy applications

- NEPTUNE will require some coupling between Fluid and Particle models
- BOUT++/Nektar/EPOCH/Firedrake are likely too large to evaluate for portability
- A number of mini-applications exist in a variety of performance portable implementations (or are simple enough that they can be ported) that we can leverage to evaluate these approaches
  - FEM/Spectral
    - MiniFE, Nekbone
  - PIC
    - MinEPOCH, VPIC, PICSARlite
  - Suggestions on a postcard…

© Crown Copyright 2021