ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

A Simple Asynchronous I/O Server

National Supercomputing Service
Swiss National Supercomputing Centre

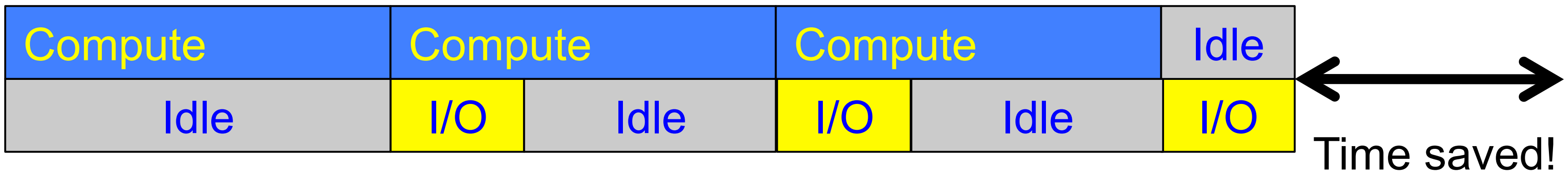
What is an asynchronous I/O server?

- In a 'normal' parallel application, I/O is handled the by same MPI tasks that handle computation.
  - Typically $ntasks_{io} <= ntasks_{compute}$
- Problem: Even if you parallelize your I/O, you still could slow down your computations a lot.
  - "Overlap communications and computations"
  - "Overlap computation and I/O".

Why use async IO? Save time!

Typically: Compute tasks are IO tasks and I/O blocks computing

With asynchronous I/O, compute tasks and IO tasks are separate



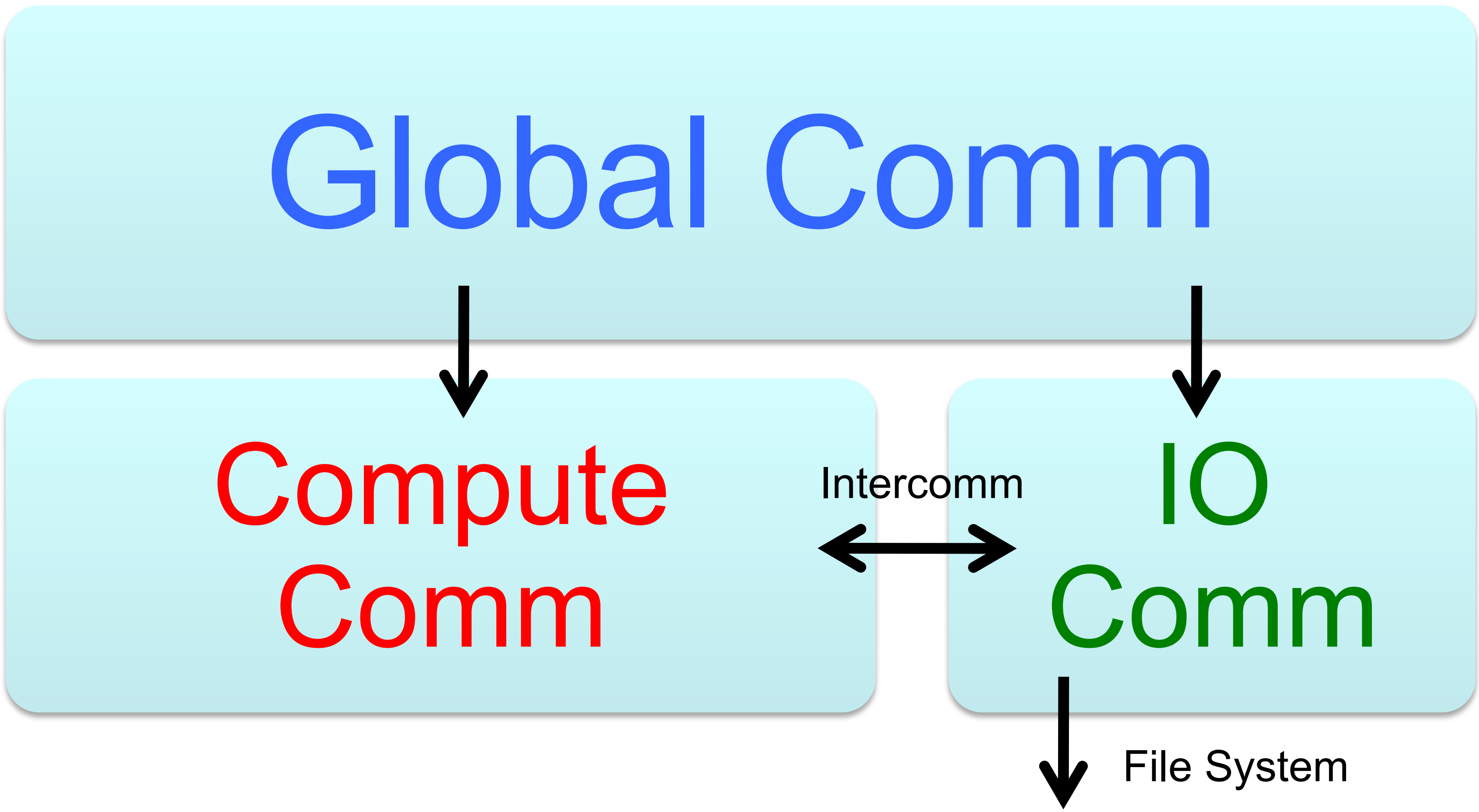
Idea: I/O tasks are idle until compute tasks send them data.
Compute tasks send data and continue
I/O tasks process and write data

Outline of an AsyncIO server

- Initialize MPI.
- Split communicators.
- Set up inter-communicator.
- Send data from compute tasks to I/O tasks through inter-communicator.
- I/O Routine waits for messages, buffers data, creates file, writes to disk.

Setting up communicators

```
CALL MPI_INIT(ierr)

CALL MPI_COMM_DUP( MPI_COMM_WORLD, globalComm, ierr )

! Determine which ranks are compute tasks and which are I/O tasks
! If compute task, set color = 1
! If I/O task, set color = 0
! Assign one task per node to be an I/O task, round robin if necessary

CALL MPI_COMM_SPLIT( globalComm, color, myrank, splitComm, ierr )

IF( compute_task )THEN
    CALL MPI_COMM_DUP( splitComm, computeComm, ierr )

IF( io_task   )THEN
    CALL MPI_COMM_DUP( splitComm, ioServeComm, ierr )
```


Global Comm
Compute Comm
Intercomm
IO Comm
File System

Setting up intercommunicator

```
!  Create an intercommunicator between the compute comm and the IO comm.
! This allows us to send data from the compute world to the I/O world using
! MPI sends and receives.

IF( compute_task )THEN
     CALL MPI_INTERCOMM_CREATE( computeComm,    0, globalComm,
                                io_start, 0, interComm, ierr )

IF( io_task )THEN
     CALL MPI_INTERCOMM_CREATE( ioservComm, 0, globalComm,
                                comp_start, 0, interComm, ierr )

! Assign compute tasks to I/O tasks. Thus 12 compute tasks assigned to 4 I/O
! tasks means each I/O task receives data from 3 compute tasks.
```

Basic job control loop

```
WHILE( notDone )

      IF( compute_task ) DO_WORK()

      IF( io_task      ) DO_IO()

END WHILE
```

Compute tasks will do some work for every iteration.

I/O tasks will wait in DO_IO() until signaled to write a file.

IO Server

```
SUBROUTINE DO_IO()

! Send my data to the I/O server, note non-blocking send
IF( compute_task )THEN
      CALL MPI_ISEND( ….,my_task_data…., myIORank, …., interComm, … )
      RETURN
END IF

! Check to see if any data has arrived, if not, then wait.
! The probe function will proceed as soon as the first message is ready

CALL MPI_PROBE( MPI_ANY_SOURCE, MPI_ANY_TAG, interComm, … )

! Use netCDF/HDF5/ADIOS API to create a file, file variables, and metadata

status = ParallelFileCreate( ioServeComm, …. )
```

IO Server

```
! Loop over the number of compute tasks that I have to get messages from
DO I = 1, numComputeTasksToRecieve

    CALL MPI_RECV( buffer, … MPI_ANY_SOURCE, MPI_ANY_TAG,
                       interComm, statuses, …)
    ! Get the rank in the compute world that sent the data
    rcvRank = statuses(MPI_SOURCE)

    ! Use rcvRank to figure out offsets and counts to place local data into global
    ! Data structure in file, e.g.

    starts(:) = (/ix,iy,1/)
    counts(:) = (/nx,ny,nz/)

    status = ParallelPutToFile( filePtr, varid, data, start=starts, count=counts )
END DO
```

Final Thoughts

- For good performance, make sure you are using collective operations.
  - Don't forget striping if you're on Lustre!
- For robustness, need to check to make sure that you've finished writing one file before you try to start writing another.
  - Could happen if compute time is less than I/O time.
- Questions?