MPI Progress-Independent Communicators

Shared Object Semantics in MPI

- Current semantics allow any thread to access any MPI object
  - Request can be created by one thread and used by another thread
  - MPI implementation requires appropriate locking/memory consistency to make sure this is allowed
- Some applications might not require such semantics
  - Each thread only uses objects generated by it

```
P0 (Thread 0)
    MPI_Irecv(…, comm1, &req1);
    pthread_barrier();

    pthread_barrier();
    MPI_Wait(&req1, …);
```

```
P0 (Thread 1)
MPI_Irecv(…, comm2, &req2);
pthread_barrier();
MPI_Wait(&req2, …);
pthread_barrier();
```

```
P1
MPI_Ssend(…, comm1);
MPI_Ssend(…, comm2);
```

Communicator Hints

- Predefined info arguments for MPI_Comm_dup_with_info
- independent_comm
  - "This info argument allows a high-quality MPI implementation to assign independent communication resources to this communicator. A thread waiting on an operation issued on a communicator with this info argument set, will not be required to make progress on pending operations on any other communicator, window or file."
  - The MPI implementation can create independent communication resources for this communicator (out-of-band communication, lesser lock contention)
- The following program, which is valid for MPI-3, will not be valid with this info hint:

P0 (Thread 0)	P0 (Thread 1)	P1
MPI_Irecv(…, comm1, &req1);	MPI_Irecv(…, comm2, &req2);	MPI_Ssend(…, comm1);
pthread_barrier();	pthread_barrier();	MPI_Ssend(…, comm2);
	MPI_Wait(&req2, …);	
pthread_barrier();	pthread_barrier();	
MPI_Wait(&req1, …);		

Other notes (1/2)

- How the semantics are inherited to other objects needs to be discussed
  - What happens when I dup an "independent" communicator
    - The standard defines that dup will inherit the info, so the new communicator will be independent as well
  - What happens when I split an "independent" communicator
    - The standard defines that split, etc., will not inherit the info, so the new communicator will not be independent
  - What happens to files, windows, etc.
    - No additional propagation of info. We can define the info key for those objects if needed

Other notes (2/2)

- Additional hints can improve performance further
  - E.g., this communicator will be used by only one thread
- Implementation details
  - Would be useful if the MPI implementation can create communicator-specific objects
  - Already done as research papers, but not in production implementations today (?)

What are we losing?

- If applications expect progress on one communicator will make progress on everything, that might not happen any more
  - Is that breaking backward compatibility?
    - Might be OK since this is a new info key
  - E.g., asynchronous progress threads