Scalable MPI Completion

9TH JLESC Workshop

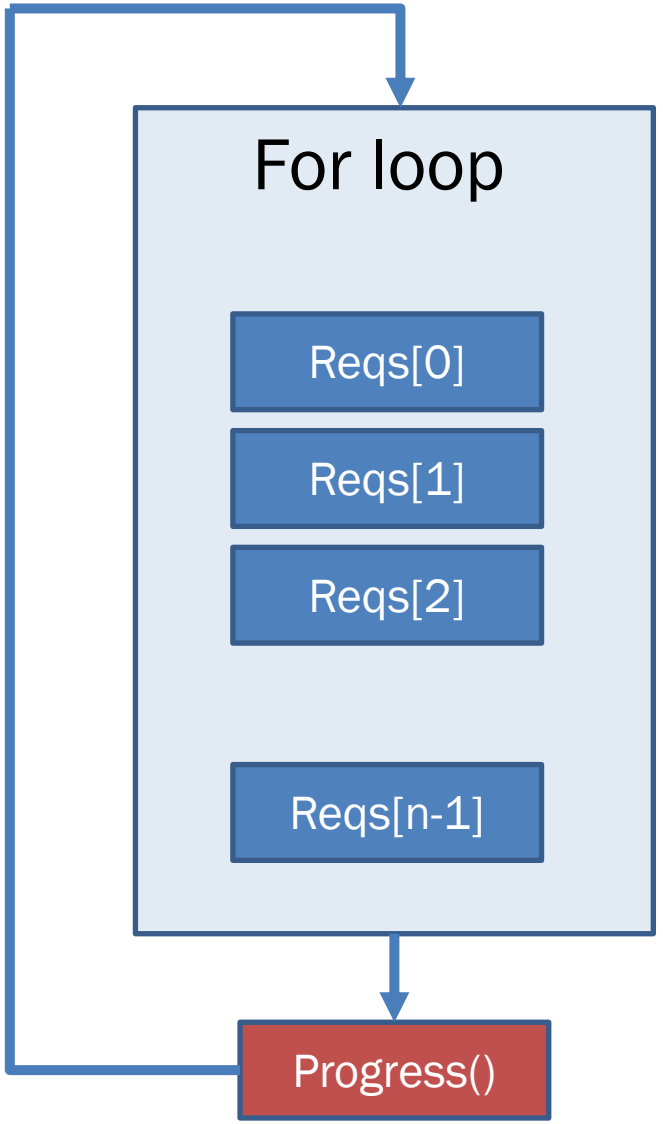
04/15/2019

Thananon "**Arm**" Patinyasakdikul

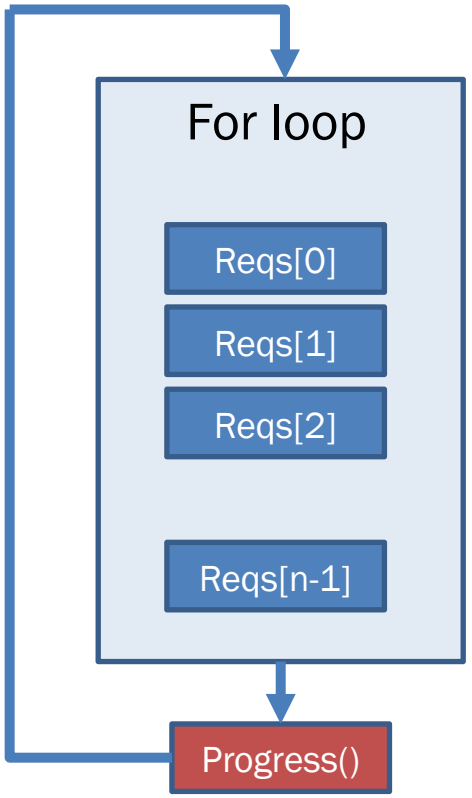
MPI Completion - Design

MPI_Waitall, MPI_Waitsome, MPI_Waitany
MPI_Testall, MPI_Testsome, MPI_Testany

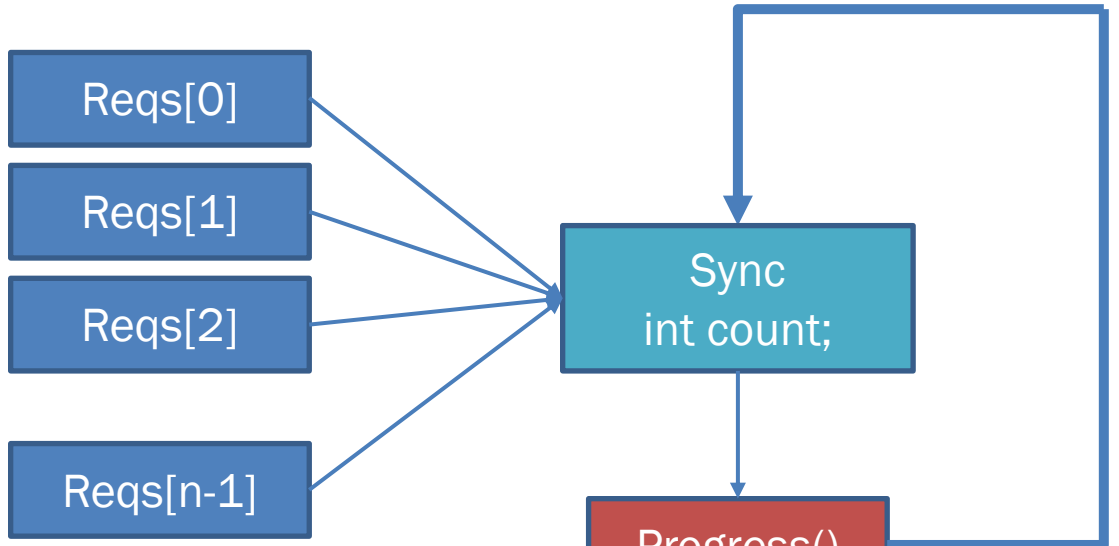
```
MPI_Isend(.., reqs[0]);
MPI_Isend(.., reqs[1]);
MPI_Isend(..,reqs[2]);
..
..
MPI_Isend(..,reqs[n-1]);

MPI_Waitall(n, &reqs, &statuses);
```



MPI Completion – Sync object



Previous OMPI implementation

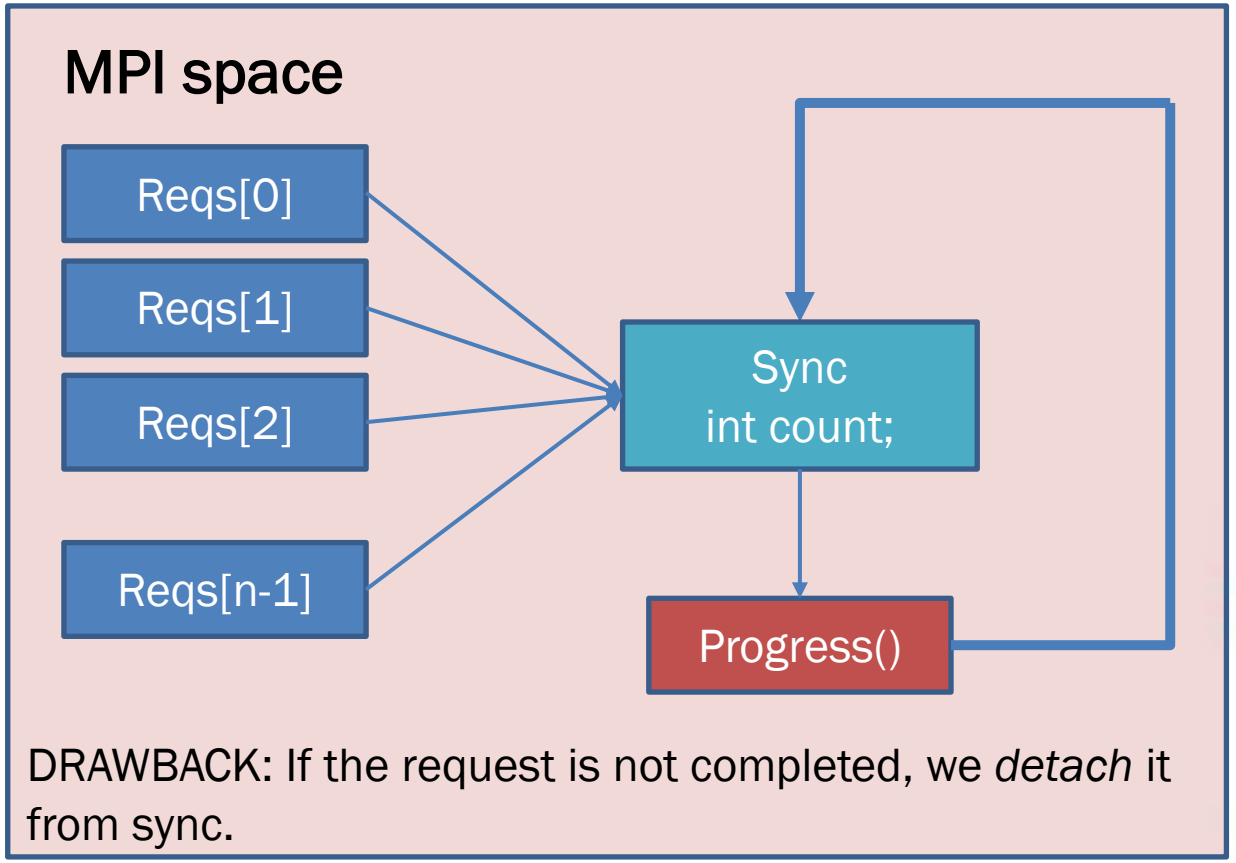


- O(1) checking
- Multithreaded benefit*

Current OMPI implementation

MPI Completion – Bad case

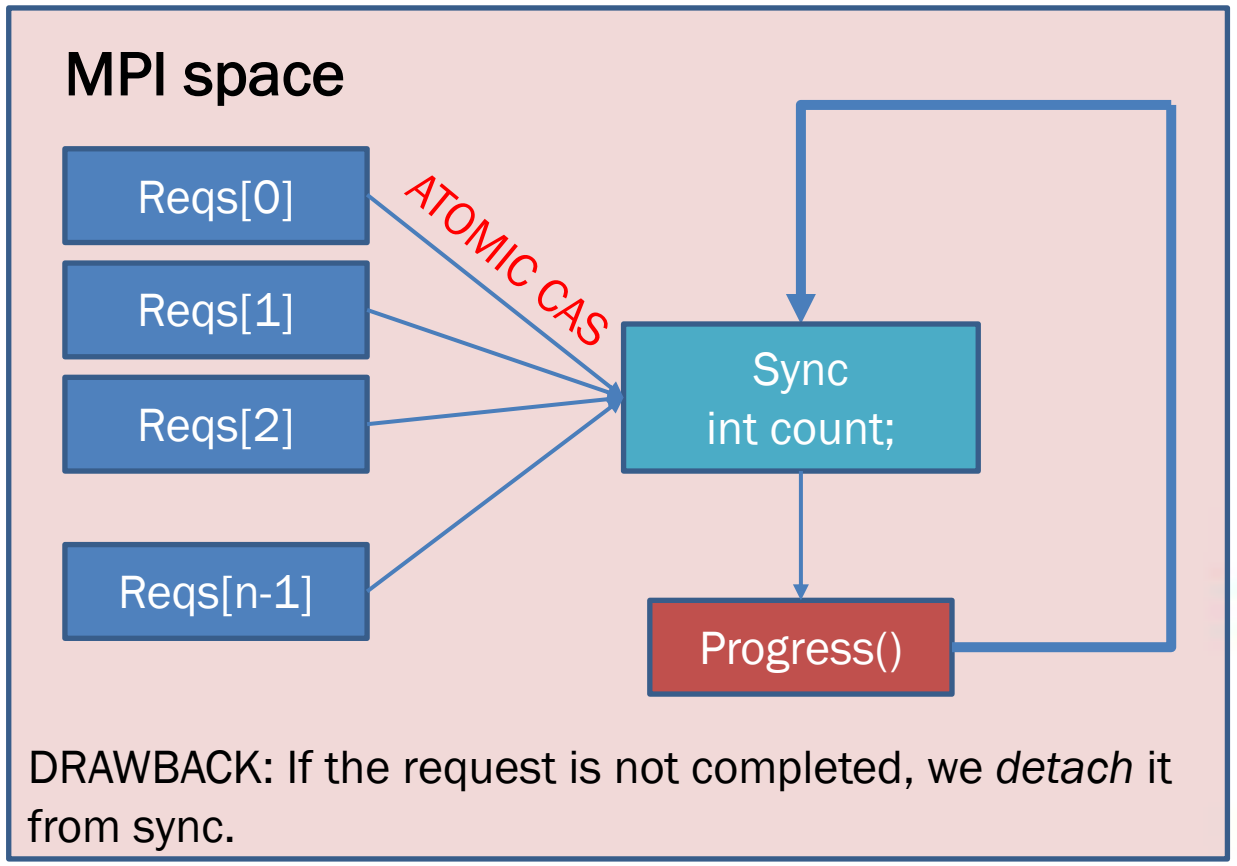
```
MPI_Isend(.., reqs[0]);
MPI_Isend(.., reqs[1]);
MPI_Isend(..,reqs[2]);
..
..
MPI_Isend(..,reqs[n-1]);

MPI_Waitsome(n, &reqs, &statuses);
...
MPI_Waitsome(n, &reqs, &statuses);
```



DRAWBACK: If the request is not completed, we *detach* it from sync.

MPI Completion – Worst case

```
MPI_Init_thread(MULTIPLE);
MPI_Isend(.., reqs[0]);
MPI_Isend(.., reqs[1]);
MPI_Isend(..,reqs[2]);
..
..
MPI_Isend(..,reqs[n-1]);

MPI_Waitsome(n, &reqs, &statuses);
...
MPI_Waitsome(n, &reqs, &statuses);
```

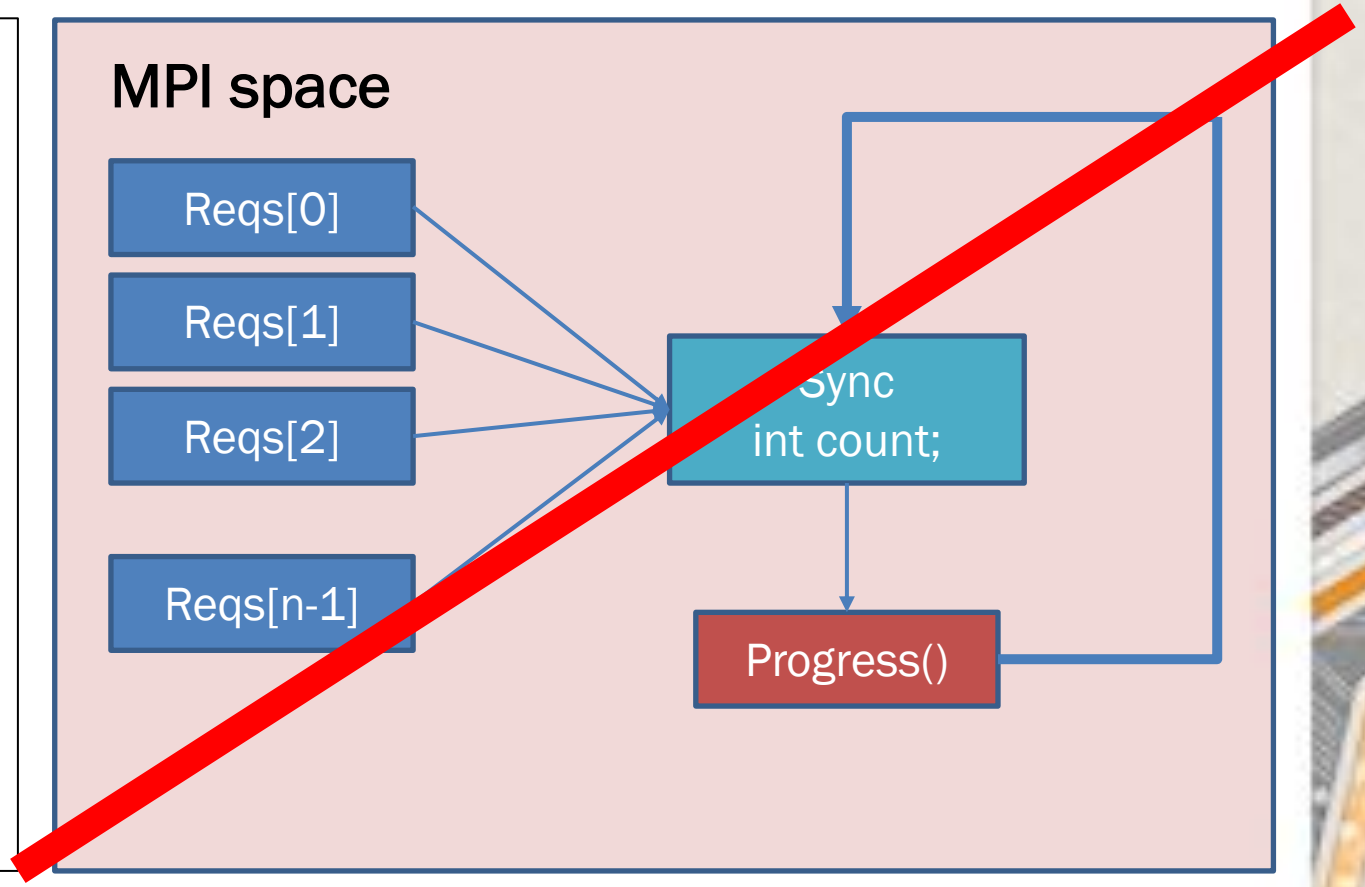


DRAWBACK: If the request is not completed, we *detach* it from sync.

MPI Completion – Not for MPI_Test*

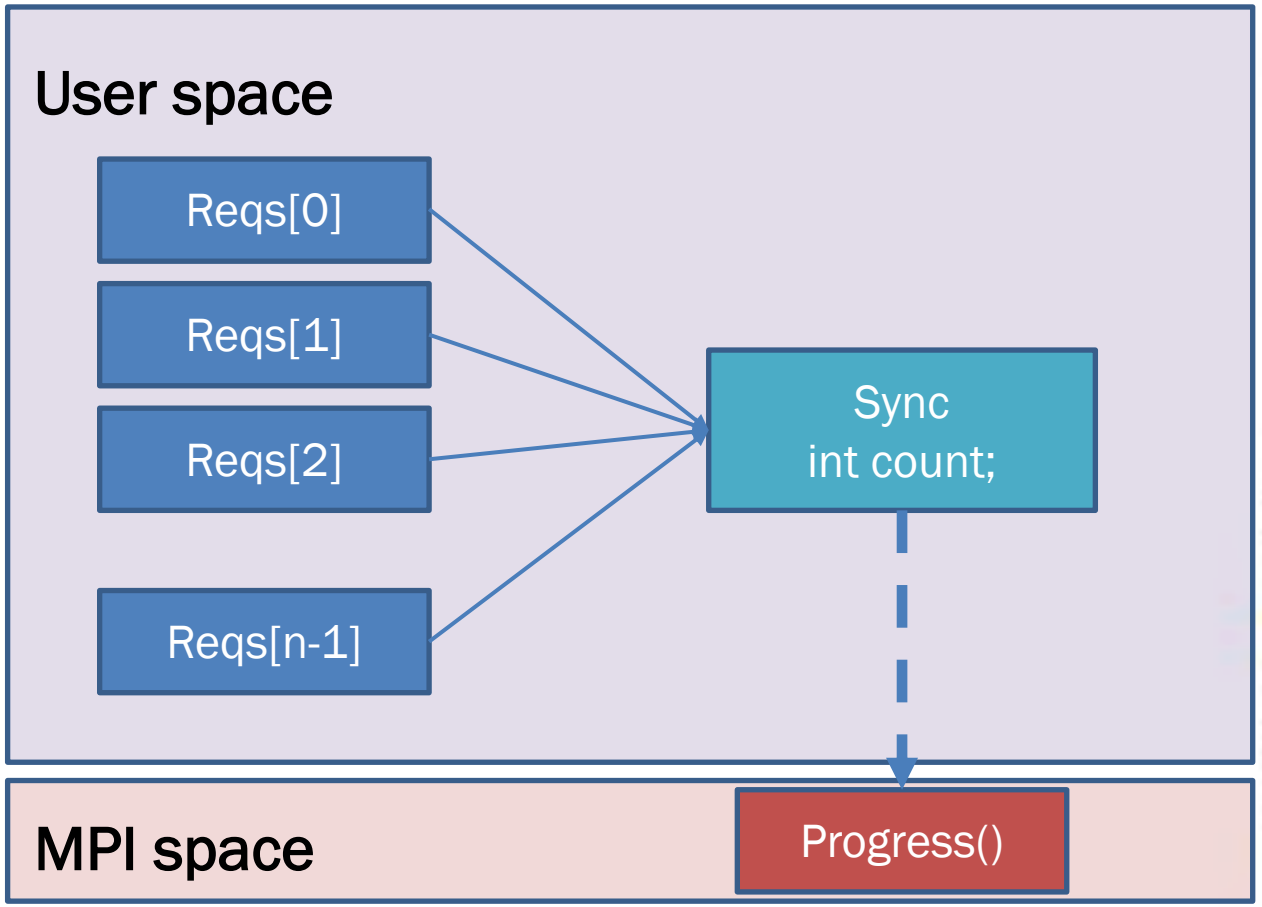
*too costly to create an object for single use.

```
MPI_Isend(.., reqs[0]);
MPI_Isend(.., reqs[1]);
MPI_Isend(..,reqs[2]);
..
..
MPI_Isend(..,reqs[n-1]);

MPI_Testsome(n, &reqs, &statuses);
...
MPI_Testsome(n, &reqs, &statuses);
```



MPIX_Sync- Design

- Give users control of the sync object.
  - Reusable (No need for array rebuilding)
  - Relieves MPI from detaching.
  - Support user-level callback function*.



MPIX_Sync- API

```
int MPIX_Sync_init(MPIX_Sync *sync);

int MPIX_Sync_attach(
        MPIX_Sync sync,
        MPI_Request request,
        void *completion_data);

void* MPIX_Sync_query(
        MPIX_Sync sync,
        MPI_Status *status);

int MPIX_Sync_query_bulk(
        int incount;
        MPIX_Sync sync,
        int *outcount;
        void **completion_data
        MPI_Status *status);
```

Initialize the object.

Attach the request with sync object and *completion data*.

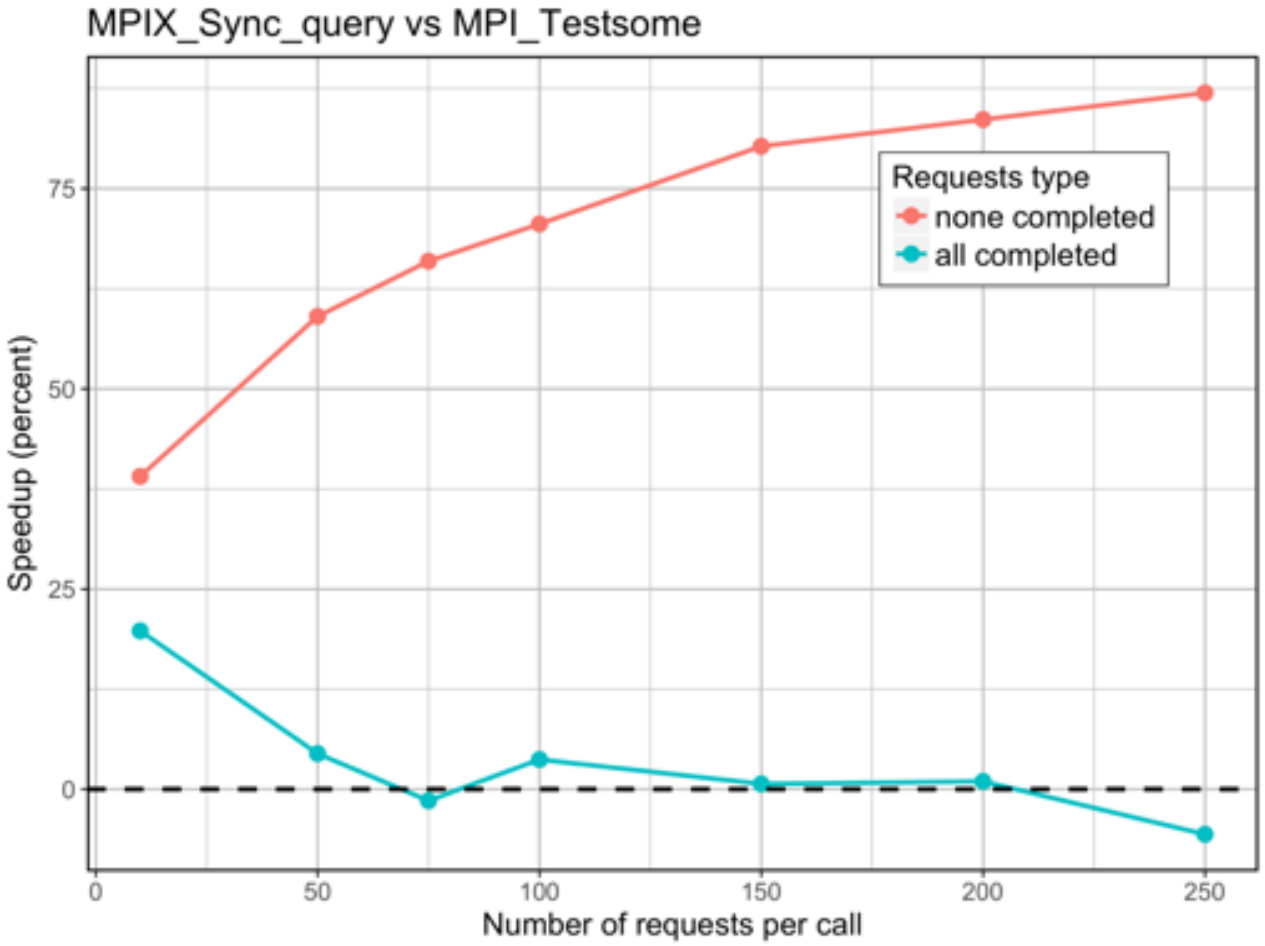
Return the first *completion_data* associated with sync object.

Return n *completion_data* associated with sync object. (testsome API)

MPIX_Sync- API

int MPIX_Sync_waitall(MPIX_Sync sync); *not sure if needed.	Wait for all attached requests to complete.
int MPIX_Sync_testall(MPIX_Sync sync); *not sure if needed.	Check if all attached requests are completed.
int MPIX_Sync_size(MPIX_Sync sync);	Get the number of incomplete requests.
int MPIX_Sync_probe(MPIX_Sync sync);	Get the number of *unconsumed* completed requests.
int MPIX_Sync_free(MPIX Sync *sync);	

MPIX_Sync_query – speedup



- Faster turnaround time if nothing completed ~ O(1).
- Slightly slower if every request completed. (Tuning possible)
- MPIX_Sync_query_bulk gives slightly better performance.
- API provides more flexibilities.

MPIX_Sync_query – Application

*prelim results from PaRSEC (Runtime task scheduler)
- ~10-30 persistent requests + MPI_Testsome()

Pingpong (communication only)

message size (bytes)	speedup
400	13.73 %
4000	7.15 %
40000	3.1 %
400000	2.55 %

DPLASMA (matrix factorization)

kernel	speedup
dpotrf	~0%*
dgeqrf	~0%*

*within error margin

MPIX_Sync_query – User level callback

```
int MPIX_Sync_init(MPIX_Sync *sync);

int MPIX_Sync_attach(
        MPIX_Sync sync,
        MPI_Request request,
        void *completion_data);

void* MPIX_Sync_query(
        MPIX_Sync sync,
        MPI_Status *status);

int MPIX_Sync_query_bulk(
        int incount;
        MPIX_Sync sync,
        int *outcount;
        void **completion_data
        MPI_Status *status);
```

Discussion:

- Users can already call the function after query. (Passing a structure as *completion_data*)
  *more control to the user

- MPI can call callback function as soon as ..
  - A request is completed
  - Certain number of requests are completed.

  *might affect communication performance

MPIX_Sync- API

```
int MPIX_Sync_init(MPIX_Sync *sync);

int MPIX_Sync_attach(
        MPIX_Sync sync,
        MPI_Request request,
        void *completion_data);

void* MPIX_Sync_query(
        MPIX_Sync sync,
        MPI_Status *status);

int MPIX_Sync_query_bulk(
        int incount;
        MPIX_Sync sync,
        int *outcount;
        void **completion_data
        MPI_Status *status);
```

- *completion_data* can be MPIX_SYNC_NO_COMPLETION_DATA.

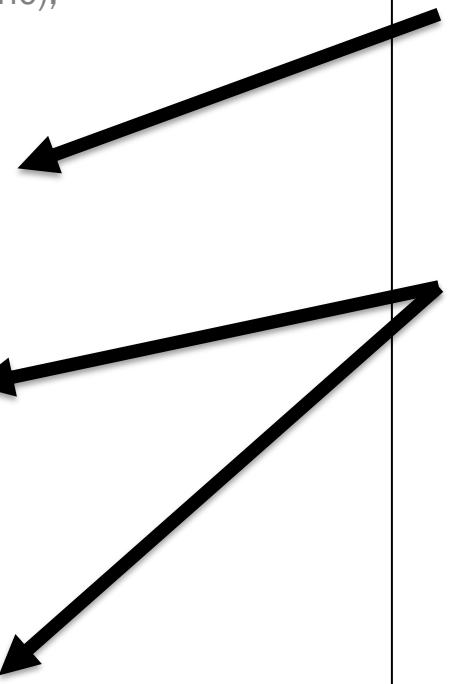
- Return MPIX_SYNC_EMPTY if there is no completion.

- *incount* can be any number.
(unlike MPI_Testsome where it has to be *nrequests*)

- Return MPIX_SYNC_MORE if there is more completion than asked for.

MPIX_Sync- API

```
int MPIX_Sync_init(MPIX_Sync *sync);

int MPIX_Sync_attach(
        MPIX_Sync sync,
        MPI_Request request,
        void *completion_data);

void* MPIX_Sync_query(
        MPIX_Sync sync,
        MPI_Status *status);

int MPIX_Sync_query_bulk(
        int incount;
        MPIX_Sync sync,
        int *outcount;
        void **completion_data
        MPI_Status *status);
```



- User relinquish the request in attach.

- We free the request here.

Users are responsible to keep track of the requests before passing it to attach if they want to *cancel* them.