MPI RMA Synchronization Asserts

All Available Asserts by Function

Function	Assert	Constant	Meaning	Optimization Impact	Risk Level
MPI_Win_fence					
	MPI_MODE_NOPRECEDE	0x1	No RMA calls before this fence since window creation or last fence	Eliminates initial barrier (~O(log p) latency)	Medium - silent corruption if violated
	MPI_MODE_NOSUCCEED	0x2	No RMA calls after this fence until window freed or next fence	Eliminates final barrier (~O(log p) latency)	Medium - silent corruption if violated
	MPI_MODE_NOSTORE	0x4	Local window not modified by local stores since last fence	Skips memory barriers/cache flush	High - data inconsistency if violated
	MPI_MODE_NOPUT	0x8	Window not target of Put/Accumulate since last fence	Skips incoming buffer processing	Medium - data corruption if violated
MPI_Win_start					
	MPI_MODE_NOCHECK	0x10	Matching MPI_Win_post already called by target processes	Eliminates synchronization with posting processes	High - deadlock/corruption if violated
MPI_Win_post					
	MPI_MODE_NOCHECK	0x10	No conflicting locks held on window	Eliminates lock conflict checking	High - race conditions if violated
	MPI_MODE_NOSTORE	0x4	Local window not modified by local stores since last PSCW sync	Skips memory synchronization	High - data inconsistency if violated
	MPI_MODE_NOPUT	0x8	Window not target of Put/Accumulate since last PSCW sync	Skips incoming operation processing	Medium - data corruption if violated
MPI_Win_complete					
	MPI_MODE_NOSTORE	0x4	Local window not modified during access epoch	Skips memory barriers	High - data inconsistency if violated
	MPI_MODE_NOPUT	0x8	Window not target of Put/Accumulate during epoch	No incoming buffer flush	Medium - ignored by most implementations
MPI_Win_wait					
	MPI_MODE_NOCHECK	0x10	MPI_Win_complete already called by all origin processes	Eliminates completion detection	High - incomplete operations if violated
MPI_Win_lock					
	MPI_MODE_NOCHECK	0x10	No conflicting locks held (shared or exclusive per lock type)	Skips lock conflict detection/resolution	High - race conditions if violated
MPI_Win_lock_all					
	MPI_MODE_NOCHECK	0x10	No conflicting locks on any target	Eliminates all lock checking overhead	High - race conditions if violated

Assert Compatibility Matrix

Assert	fence	start	post	complete	wait	lock	lock_all
MPI_MODE_NOPRECEDE	✓	x	x	x	x	x	x
MPI_MODE_NOSUCCEED	✓	x	x	x	x	x	x
MPI_MODE_NOSTORE	✓	x	✓	✓	x	x	x
MPI_MODE_NOPUT	✓	x	✓	✓	x	x	x
MPI_MODE_NOCHECK	x	✓	✓	x	✓	✓	✓

Valid Assert Combinations

Function	Common Combinations	Use Case	Performance Gain
MPI_Win_fence			
	0	General purpose (safe)	Baseline
	MPI_MODE_NOPRECEDE	First fence after window creation	~50% latency reduction
	MPI_MODE_NOSUCCEED	Last fence before window destruction	~50% latency reduction
	MPI_MODE_NOPRECEDE | MPI_MODE_NOSUCCEED	Single epoch pattern	~75% latency reduction
	MPI_MODE_NOSTORE | MPI_MODE_NOPUT	Read-only epoch	Memory barrier elimination
	MPI_MODE_NOPRECEDE | MPI_MODE_NOSTORE | MPI_MODE_NOPUT | MPI_MODE_NOSUCCEED	Initialization fence	Maximum optimization
MPI_Win_start			
	0	Safe synchronization	Baseline
	MPI_MODE_NOCHECK	Known posting complete	Eliminates barrier
MPI_Win_post			
	0	General purpose	Baseline
	MPI_MODE_NOCHECK	No lock conflicts guaranteed	Skip conflict detection

Function	Common Combinations	Use Case	Performance Gain
	MPI_MODE_NOSTORE	No local modifications	Skip memory sync
	MPI_MODE_NOCHECK | MPI_MODE_NOSTORE	Passive target optimization	Maximum for post
MPI_Win_lock			
	0	Safe locking	Full checking
	MPI_MODE_NOCHECK	Application-level synchronization ensures no conflicts	Skip lock acquisition protocol

Assert Semantic Requirements

Assert	Must Guarantee	Collective?	Typical Violation Symptoms
NOPRECEDE	No process issued RMA ops before fence	Yes - all processes	Silent data corruption, missing updates
NOSUCCEED	No process will issue RMA ops after fence	Yes - all processes	Deadlock, incomplete operations
NOSTORE	Process didn't modify window memory locally	Process-local*	Stale data, memory inconsistency
NOPUT	Window wasn't target of Put/Accumulate	Process-local*	Lost updates, data corruption
NOCHECK	Synchronization already established externally	Depends on context	Deadlock, race conditions

*Note: While NOSTORE/NOPUT are process-local conditions, in fence they must be asserted collectively by all processes

Performance Impact Summary

Synchronization Mode	Assert	Latency Saved	Throughput Impact	Memory Saved
Fence				
	NOPRECEDE	$\alpha \log p$	None	None
	NOSUCCEED	$\alpha \log p$	None	None

Synchronization Mode	Assert	Latency Saved	Throughput Impact	Memory Saved
	NOSTORE	~100-1000 ns	None	Cache flush overhead
	NOPUT	O(queue_size)	~10-20% for small msgs	Buffer management
PSCW				
	NOCHECK (start)	α log g	None	None
	NOCHECK (post)	Lock check time	None	Lock tracking structures
	NOSTORE	Memory fence time	None	Cache coherency traffic
Lock				
	NOCHECK	Lock acquisition	~2-3x for contended	Lock queue management

Where: α = network latency, p = total processes, g = group size

Usage Examples with Multiple Asserts

```
// Fence: First operation after window creation
MPI_Win_fence(MPI_MODE_NOPRECEDE | MPI_MODE_NOSTORE, win);
// Can combine because:
// - NOPRECEDE: No RMA before (just created)
// - NOSTORE: No local stores yet

// Fence: Last operation before freeing
MPI_Win_fence(MPI_MODE_NOSUCCEED | MPI_MODE_NOPUT, win);
// Can combine because:
// - NOSUCCEED: No more RMA (about to free)
// - NOPUT: Know no incoming Puts in this pattern

// PSCW: Optimized post for passive target
MPI_Win_post(group, MPI_MODE_NOCHECK | MPI_MODE_NOSTORE, win);
// Can combine because:
// - NOCHECK: No locks active
// - NOSTORE: Haven't modified local window

// Lock: Application-managed synchronization
MPI_Win_lock(MPI_LOCK_EXCLUSIVE, target, MPI_MODE_NOCHECK, win);
// Use when:
// - External sync ensures no conflicts
// - E.g., token-based or barrier-based coordination
```

Warning Matrix

Function	Assert	Never Use If	Common Mistake
fence	NOPRECEDE	Any process did RMA after last fence	Using after any RMA operation
	NOSUCCEED	Any process needs RMA before next fence	Using when more communication needed
	NOSTORE	Modified window memory locally	Confusing with "no Put operations"
	NOPUT	Others might Put/Accumulate to you	Assuming Get operations don't count
start	NOCHECK	Post might not be called yet	Racing Post/Start calls
post	NOCHECK	Locks might be held	Not tracking all lock usage
lock	NOCHECK	Other locks might exist	Assuming shared locks don't conflict

Implementation-Specific Assert Behavior

Implementation	Strictly Enforces	Optimizes	Ignores	Debug Mode Check
Intel MPI	NOCHECK	NOPRECEDE, NOSUCCEED	NOSTORE on unified	Yes with I_MPI_DEBUG
Open MPI	All in debug	NOPRECEDE, NOSUCCEED, NOCHECK	NOPUT often	Yes with --enable-debug
MPICH	NOCHECK	All	None	Yes with --enable-g=all
Cray MPI	All	All	None	Yes with MPICH_DBG
IBM Spectrum	NOCHECK	NOPRECEDE, NOSUCCEED	NOSTORE on Power9	Yes with MP_EUIDEVELOP

Decision Flow Chart

```
Choose Assert:
├─ MPI_Win_fence
│  ├─ First fence after create? → ADD NOPRECEDE
│  ├─ Last fence before free? → ADD NOSUCCEED
│  ├─ No local stores? → ADD NOSTORE
│  └─ No incoming Puts? → ADD NOPUT
```

```
├─ MPI_Win_start
│  └─ Know Post completed? → USE NOCHECK
├─ MPI_Win_post
│  ├─ No active locks? → ADD NOCHECK
│  └─ No local stores? → ADD NOSTORE
├─ MPI_Win_lock
│  └─ External sync guarantees no conflicts? → USE NOCHECK
└─ None certain? → USE 0 (no asserts)
```

Quick Reference

Goal	Function	Assert to Use
Skip initial barrier	fence	MPI_MODE_NOPRECEDE
Skip final barrier	fence	MPI_MODE_NOSUCCEED
Skip memory sync	fence/post/complete	MPI_MODE_NOSTORE
Skip Put processing	fence/post/complete	MPI_MODE_NOPUT
Skip sync check	start/post/wait/lock	MPI_MODE_NOCHECK
Maximum fence optimization	fence	NOPRECEDE | NOSUCCEED | NOSTORE | NOPUT
Safe default	any	0

Notes

1. **Collective Requirements**: Fence asserts must be identical across all processes
2. **Undefined Behavior**: Incorrect asserts cause silent corruption, not errors
3. **Debugging**: Always test with asserts=0 first, add optimizations incrementally
4. **Portability**: Not all implementations optimize all asserts equally
5. **Composition**: Library code should never use asserts without documenting requirements