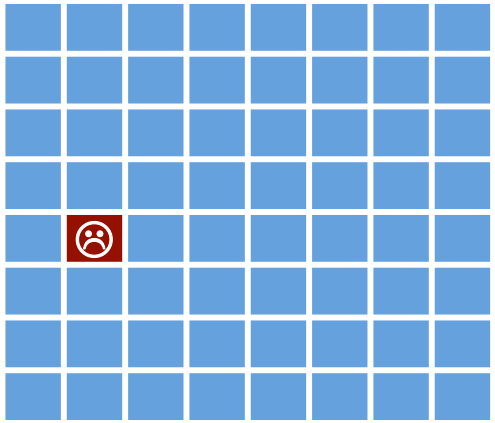
Group-Collective Communicator Creation

Ticket #286

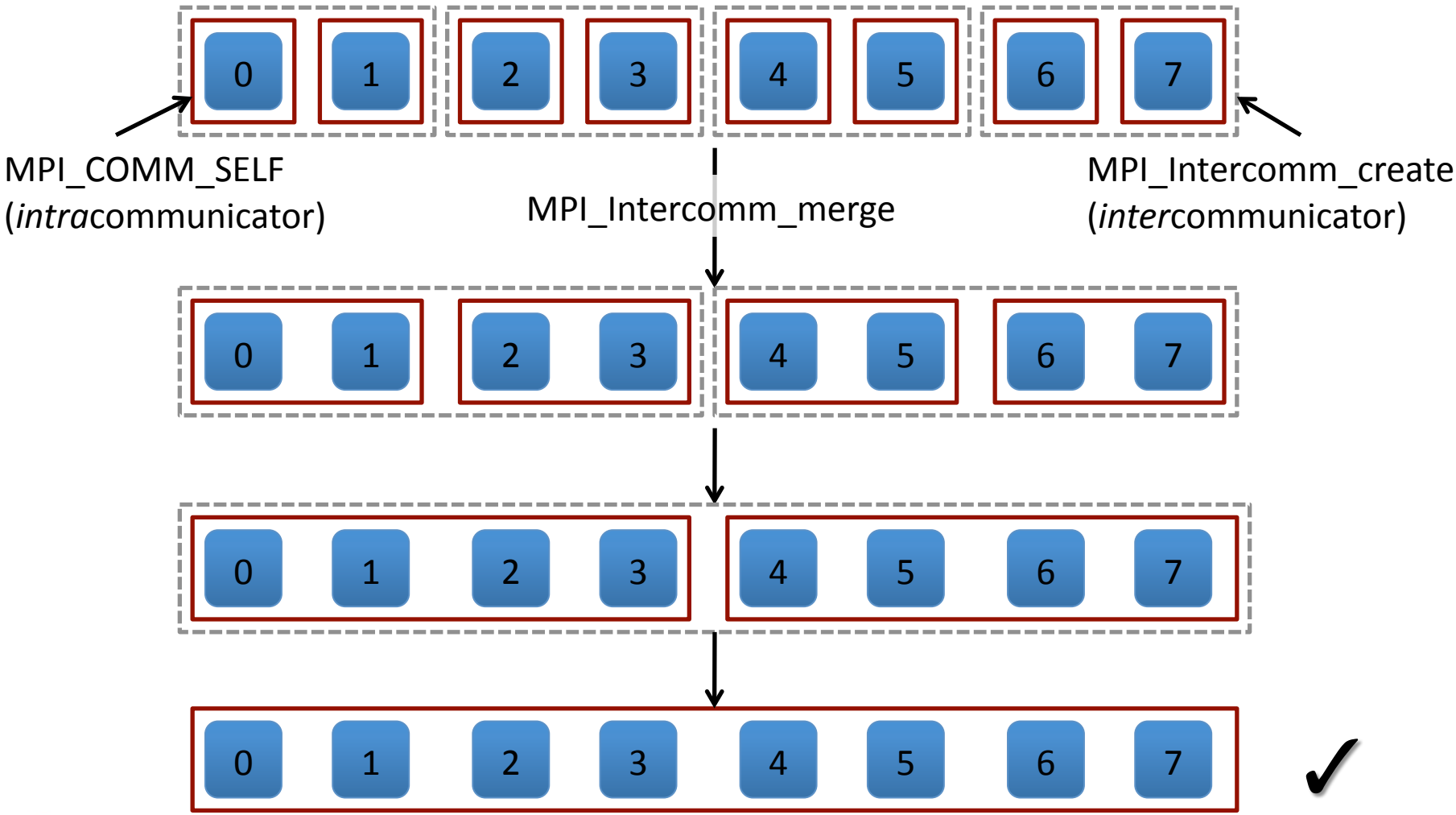
Non-Collective Communicator Creation in MPI. Dinan, et al., Euro MPI '11.

Non-Collective Communicator Creation

- Create a communicator collectively only on new members
- Global Arrays process groups
  - Past: collectives using MPI Send/Recv
- Overhead reduction
  - Multi-level parallelism
  - Small communicators when parent is large
- Recovery from failures
  - Not all ranks in parent can participate
- Load balancing
  - Reassign processes from idle groups to active groups



Non-Collective Communicator Creation Algorithm



Non-Collective Algorithm in Detail

INPUT: $group$, $comm$, tag
OUTPUT: $comm'$
REQUIRE: $group$ is ordered by desired rank in $comm'$ and is identical on all callers
LET: $grp\_pids[0..|group|-1] = \mathbb{N}$ and $pids[\,]$ be arrays of length $|group|$

MPI_Comm_rank($comm$, $\&rank$)
MPI_Group_rank($group$, $\&grp\_rank$), MPI_Group_size($group$, $\&grp\_size$)
MPI_Comm_dup(MPI_COMM_SELF, $\&comm'$)

MPI_Comm_group($comm$, $\&parent\_grp$)
MPI_Group_translate_ranks($group$, grp_size, grp_pids, $parent\_grp$, $pids$)
MPI_Group_free($\&parent\_grp$)

Translate group ranks to ordered list of ranks on parent communicator

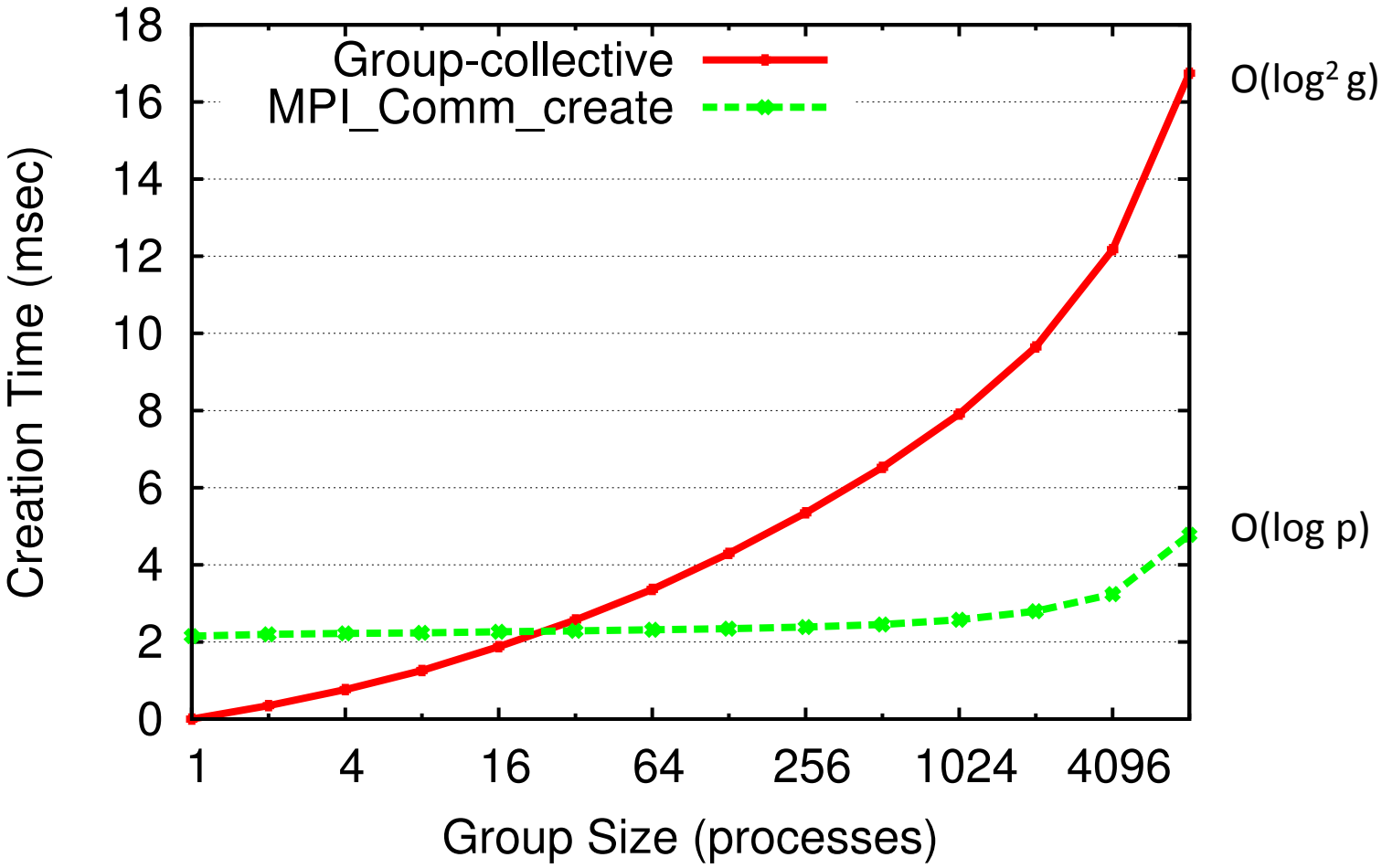
for ($merge\_sz \leftarrow 1$; $merge\_sz < grp\_size$; $merge\_sz \leftarrow merge\_sz \cdot 2$) **do**
  $gid \leftarrow grp\_rank / merge\_sz$, $comm\_old \leftarrow comm'$
  if $gid \mod 2 = 0$ **then**
    if $((gid+1) \cdot merge\_sz < grp\_size$ **then**
      MPI_Intercomm_create($comm'$, 0, $comm$, $pids[(gid+1) \cdot merge\_sz]$, tag, $\&ic$)
      MPI_Intercomm_merge(ic, 0 /* LOW */, $\&comm'$)
    end if
  else
    MPI_Intercomm_create($comm'$, 0, $comm$, $pids[(gid-1) \cdot merge\_sz]$, tag, $\&ic$)
    MPI_Intercomm_merge(ic, 1 /* HIGH */, $\&comm'$)
  end if
  if $comm' \neq comm\_old$ **then**
    MPI_Comm_free($\&ic$)
    MPI_Comm_free($\&comm\_old$)
  end if
end for

Calculate my group ID

Left group

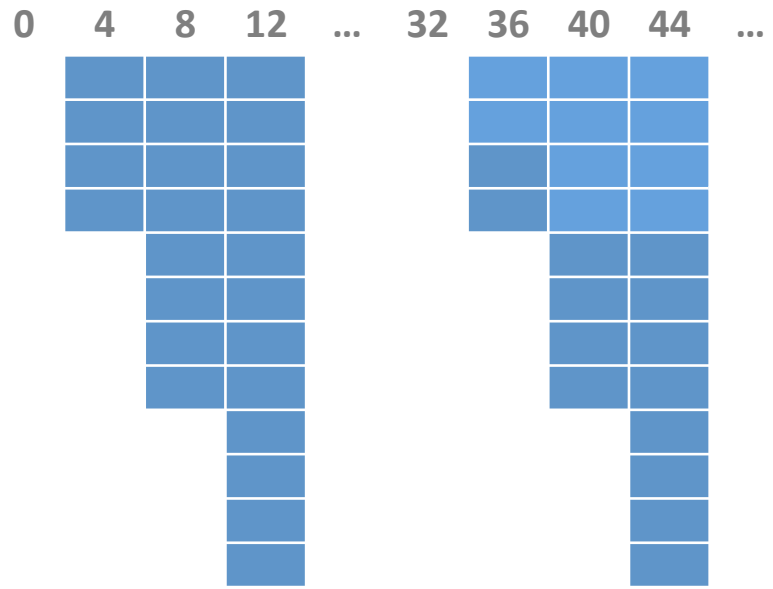
Right group

Evaluation: Microbenchmark



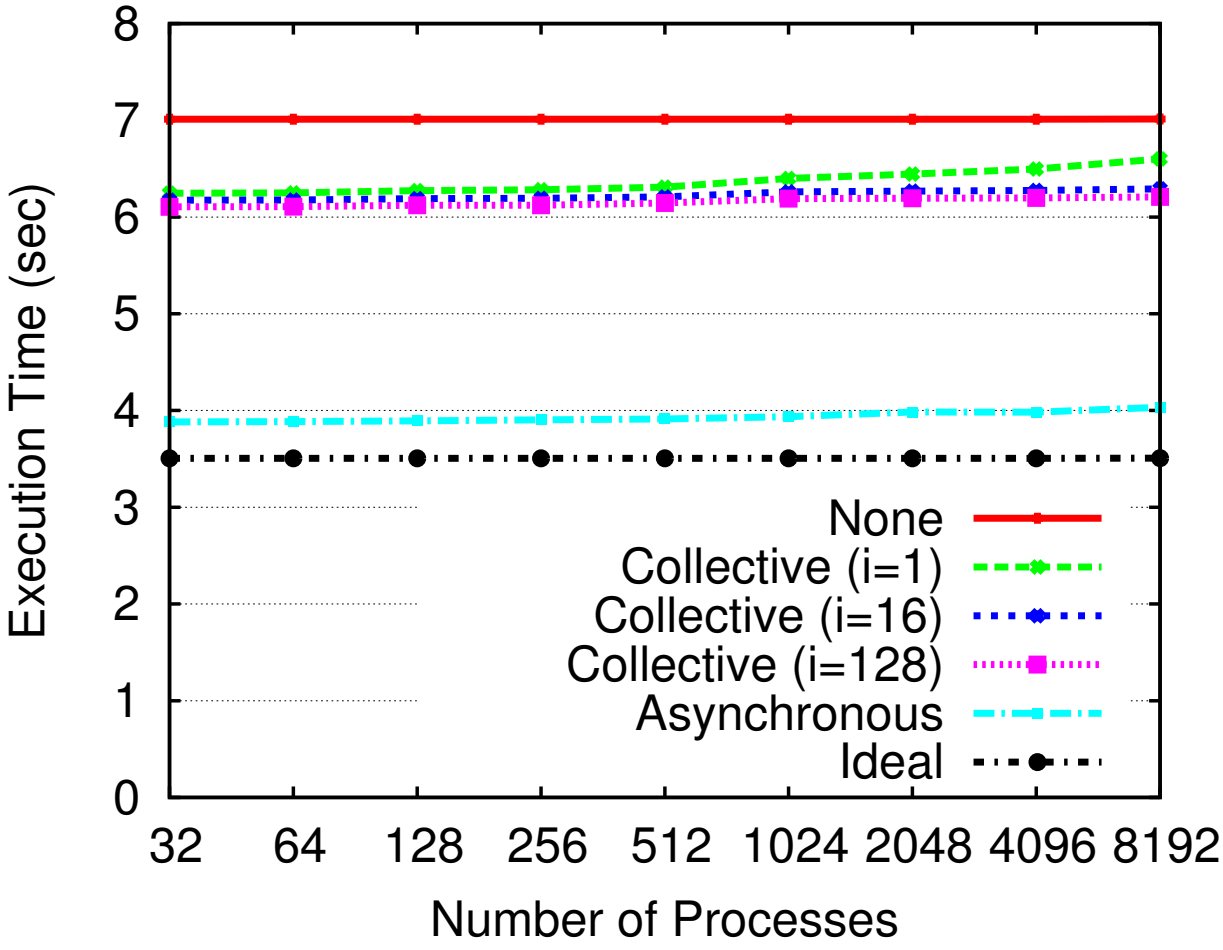
MCMC Benchmark Kernel

- Simple MPI-only benchmark
- "Walker" groups
  - Initial group size: G = 4
- Simple work distribution
  - Samples: S = (leader % 32) * 10,000
  - Sample time: t_s = 100 ms / |G|
  - Weak scaling benchmark



- Load balancing
  - Join right when idle
  - Asynchronous: Check for merge request after each work unit
  - Synchronous: Collectively regroup every *i* samples

Evaluation: Load Balancing Benchmark Results



Ticket #286: Group-collective communicator creation

MPI_Comm_create_group(comm, group, tag, newcomm)

IN	comm	communicator (handle)
IN	group	Group, which is a subset of the group of comm (handle)
IN	tag	"safe" tag (integer)
OUT	newcomm	new communicator (handle)

```
int MPI_Comm_create_group(MPI_Comm comm, MPI_Group group,
    int tag, MPI_Comm *newcomm)
```

```
MPI_COMM_CREATE(COMM, GROUP, TAG, NEWCOMM, IERROR)
    INTEGER COMM, GROUP, KEY, NEWCOMM, IERROR
```

MPI_COMM_CREATE_GROUP provides the same semantics as MPI_COMM_CREATE, however *comm* must be an intracommunicator and this routine need only be invoked by processes in the input group, *group*. Like MPI_COMM_CREATE, if a process calling MPI_COMM_CREATE_GROUP is not a member of given group it returns MPI_COMM_NULL. If different groups are provided by different callees, the calls will be interpreted as separate group-collective communicator construction operations; each call requires all members of the group to invoke this routine. This results in the same behavior as MPI_COMM_CREATE with different group arguments.

Advice to users: Group-collective creation of an intercommunicator can be achieved by creating the local communicator using MPI_COMM_CREATE_GROUP and using this as the input to MPI_INTERCOMM_CREATE.

This call may use point-to-point communication with communicator *comm*, and tag *tag* between processes in *group*. Thus, care must be taken that there be no pending communication on *comm* that could interfere with this communication. Likewise, if a given process performs multiple subset-collective communicator creation calls, the user must be careful to order calls and use different tag arguments to ensure that calls match correctly.

Advice to users: MPI_COMM_CREATE may provide lower overhead because it can take advantage of collective communication on *comm* when constructing *newcomm*.

Proceed with proposal? Yes (16), No (0), Abstain (1)