

Malleability Interface for applications

Alberto Cascajo, Javier Fernández, David E. Singh

October 14-16, 2026, Madrid, Spain

38th IEEE/SBC International Symposium on Computer Architecture and High-Performance Computing

ARCOS
COMPUTER ARCHITECTURE, COMMUNICATIONS AND SYSTEMS RESEARCH GROUP

Outline

- Use of malleable functionality
- Global malleable communicator
- Monitoring service
- Malleability region
- Attributes
  - Register/update key-values
  - Get value size and value
  - Example of attributes
- Launcher tool: mpirun.flexmpi

Use of malleable functionality

- Including malleable header/module:
  - **C:** `#include <empi.h>`
  - **F:** `use admire_wrapper`
- Wrapped MPI basic functions:
  - MPI_Init: needed to activate malleability features.

    C:
    ```
    MPI_Init(argc,argv);
    ```
    F:
    ```
    call FMPI_Init(ierror)
    ```
  - MPI_Finalize: needed to deactivate malleability features.

    C:
    ```
    MPI_Finalize();
    ```
    F:
    ```
    call FMPI_Finalize(ierror)
    ```

Global malleable communicator

- Admire Global Communicator:
  - **C:** `ADM_COMM_WORLD`
  - **F:** `call ADM_GetComm(ADM_COMM_WORLD)`
- Behaviour:
  - The Admire communicator includes all malleable processes that are active at that moment.

    Behaves like the MPI_COMM_WORLD but for malleable applications
  - The ADM_COMM_WORLD can be updated after finishing a malleable region.

    In Fortran this update must be explicit by executing:

    call ADM_GetComm(ADM_COMM_WORLD)

Monitoring service

- Activate:
  - **C:** `ADM_MonitoringService(ADM_SERVICE_START);`
  - **F:** `call ADM_MonitoringService(ADM_SERVICE_START)`
- Deactivate:
  - **C:** `ADM_MonitoringService(ADM_SERVICE_STOP);`
  - **F:** `call ADM_MonitoringService(ADM_SERVICE_STOP)`
- Behaviour:
  - Activate/deactivate the monitoring services for the malleability regions.
  - The monitoring happens only within a malleable region.

Malleability region

- Starts:
  - **C:** `ADM_MalleableRegion(ADM_SERVICE_START);`
  - **F:** `call ADM_MalleableRegion(ADM_SERVICE_START, status);`
- Ends:
  - **C:** `status = ADM_MalleableRegion(ADM_SERVICE_STOP);`
  - **F:** `call ADM_MalleableRegion(ADM_SERVICE_STOP, status);`
- Behaviour:
  - Delimits the malleability region where the monitorization is performed.
  - At the start of the region hints are sent to the IC and resources are getting booked.
  - At the end, (if resources are ready) processes are spawned or removed.
    - Status == `ADM_ACTIVE` => process remains active.
    - Status == `ADM_REMOVED` => process is removed and must end.
    - Status == `ADM_RESTART` => process is removed in order to restart it.
  - Registers the Nº of malleability regions executed (`ADM_GLOBAL_ITERATION`)
  - After the end, the rank and processes number may change and should be updated.

Attributes: Register/update for simple types

- Register/update for Integer:
  - **C:** `ADM_RegisterSysAttributesInt(”KEY", &intValue);`
  - **F:** `call ADM_RegisterSysAttributesInt(”KEY", intValue)`
- Register/update for Double:
  - **C:** `ADM_RegisterSysAttributesDouble(”KEY", &realValue);`
  - **F:**
```
call ADM_RegisterSysAttributesDouble(”KEY",
                                     realValue)
```
- Behaviour:
  - If first time, The new key is register with a copy of the value.
  - If Key already exist:
    If Value == NULL, key is removed.
    Else, removed previous value and store a copy of the new one.

Attributes: Register/update for arrays

- Register/update for Integer Arrays:
  - **C:** `ADM_RegisterSysAttributesIntArr("KEY",intArray, size);`
  - **F:**
```
call ADM_RegisterSysAttributesIntArr("KEY", intArray,
                                     size)
```
- Register/update for Double Arrays:
  - **C:**
```
ADM_RegisterSysAttributesDoubleArr("KEY", realArray,
                                   size);
```
  - **F:**
```
call ADM_RegisterSysAttributesDoubleArr("KEY",
                                        realArray, size)
```
- Behaviour:
  - If first time, The new key is register with a copy of the value.
  - If Key already exist:
    If Value == NULL, key is removed.
    Else, removed previous value and store a copy of the new one.

Attributes: Register/update for strings

- Register/update for String:
  - **C:** `ADM_RegisterSysAttributesStr(”KEY", string, size);`
  - **F:** `call ADM_RegisterSysAttributesStr(”KEY", string, size)`

- Behaviour:
  - If first time, The new key is register with a copy of the value.
  - If Key already exist:

    If Value == NULL, key is removed.

    Else, removed previous value and store a copy of the new one.

Attributes: Get value for simple types

- Get value for Integer:
  - **C:** `ADM_GetSysAttributesInt(”KEY", &intValue);`
  - **F:** `call ADM_GetSysAttributesInt(”KEY", intValue)`
- Get value for Double:
  - **C:** `ADM_GetSysAttributesDouble(”KEY", &realValue);`
  - **F:** `call ADM_GetSysAttributesDouble(”KEY", realValue)`
- Behaviour:
  - Return the value stored within the selected key
  - Value variable must be previously booked already.
  - Enough memory must be pre-allocated by the caller.

Attributes: Get value for arrays

- Get value for Integer Arrays:
  - **C:** `ADM_RegisterSysAttributesIntArr(”KEY",intArray, size);`
  - **F:** `call ADM_RegisterSysAttributesIntArr(”KEY", intArray, size)`
- Get value for Double Arrays:
  - **C:** `ADM_RegisterSysAttributesDoubleArr(”KEY", realArray, size);`
  - **F:** `call ADM_RegisterSysAttributesDoubleArr(”KEY", realArray, size)`
- Behaviour:
  - Return the value stored within the selected key
  - Value variable must be previously booked already.
  - Enough memory must be pre-allocated by the caller.

Attributes: Get value for strings

- Get value for String:
  - **C:** `ADM_GetSysAttributesStr(”KEY", string, size);`
  - **F:** `call ADM_GetSysAttributesStr(”KEY", string, size)`

- Behaviour:
  - Return the value stored within the selected key
  - Value variable must be previously booked already.
  - Enough memory must be pre-allocated by the caller.

Attributes: Examples

- Key: ADM_GLOBAL_ITERATION -> Integer
  - Sets the current iteration/malleability region that is being executed in all processes. It must be updated by the process.
- Key: ADM_GLOBAL_MAX_ITERATION -> Integer
  - Sets the maximum number of iteration/malleability region to execute for monitoring and scheduling purposes. It must be updated by the process.
- Key: ADM_GLOBAL_PROCESS_TYPE -> Integer
  - Returns whether the process has been created from the start (ADM_NATIVE) or it just has been created dynamically afterwards (ADM_SPAWNED).

Attributes: Examples

- Key: ADM_GLOBAL_HINT_NUM_PROCESS -> Integer
  - Stores a hint for the next malleability region indicating the number of processes to spawn/remove.

- Key: ADM_GLOBAL_HINT_EXCL_NODES -> Integer
  - Stores a boolean hint for the next malleability region indicating whether nodes can run exclusively one single process (or not).

Launcher tool: mpirun.flexmpi

- FlexMPI includes a modified version of the launcher tool: mpirun.flexmpi
  - Allows to launch normal, dynamic and start-stop MPI applications.
  - All options for mpirun.mpich are available as well.

- Launching normal MPI applications:

  $ mpirun.flexmpi <Application Program>

- Launching dynamic MPI applications:

  $ mpirun.flexmpi –dynamic <Application Program>

- Launching start-stop MPI applications:

  $ mpirun.flexmpi –startstop <Application Program>



Malleability Interface for applications

Alberto Cascajo, Javier Fernández, David E. Singh

October 14-16, 2026, Madrid, Spain

38th IEEE/SBC International Symposium on Computer Architecture and High-Performance Computing

ARCOS
COMPUTER ARCHITECTURE, COMMUNICATIONS AND SYSTEMS RESEARCH GROUP