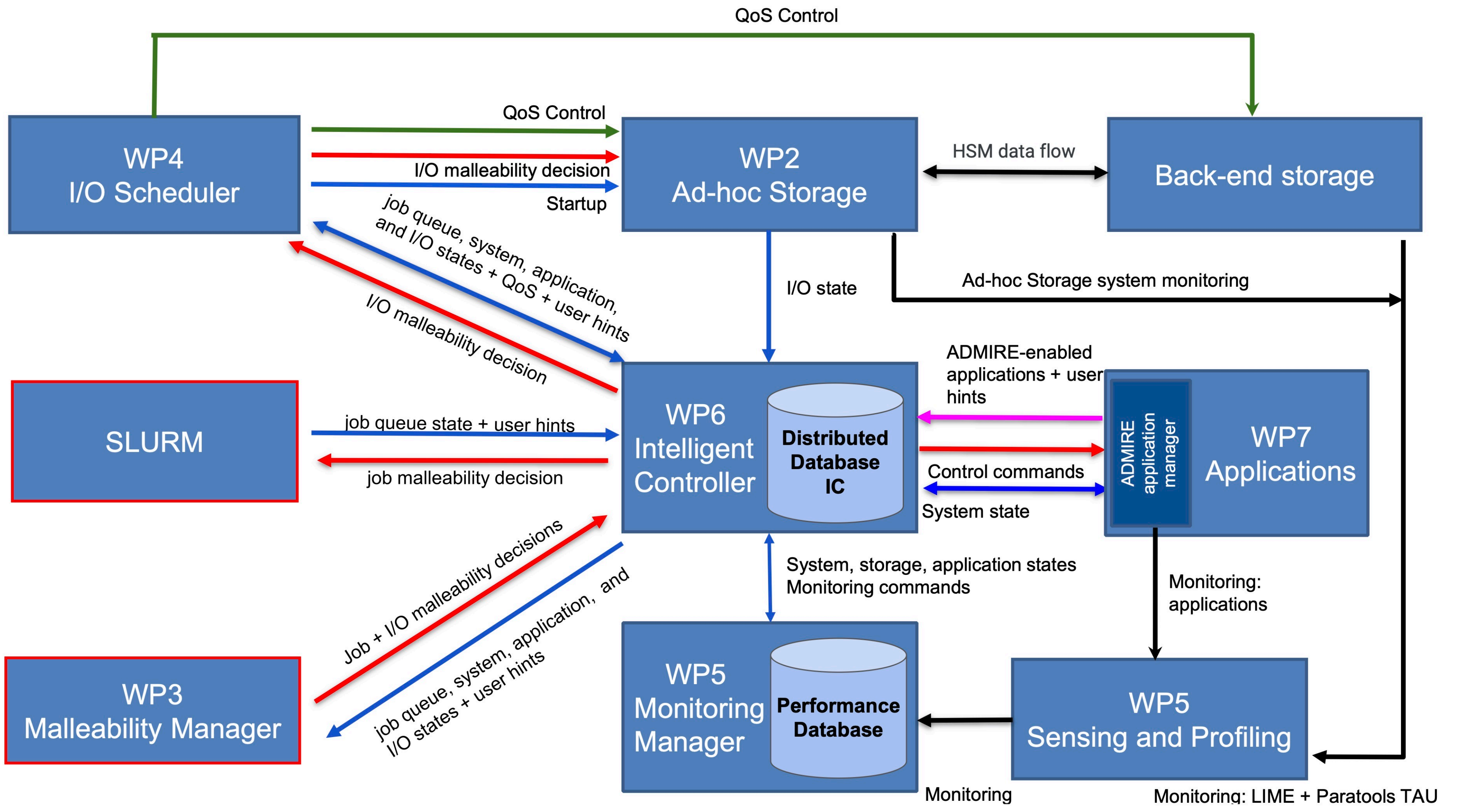
Adaptive multi-tier intelligent data manager for Exascale

ADMIRE interface for malleability

Universidad Carlos III de Madrid

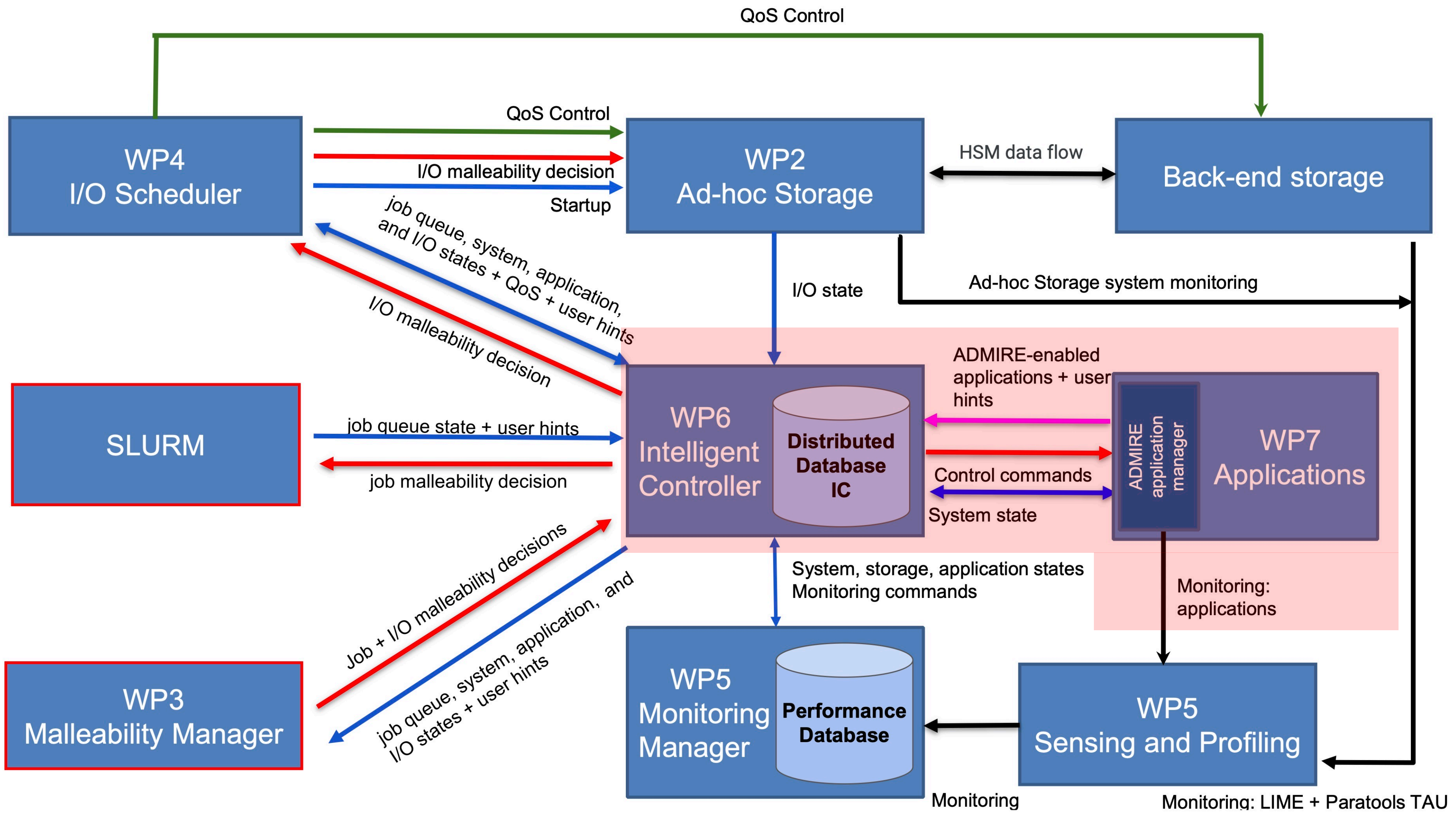
December 12th 2023

Grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1



QoS Control
WP4
I/O Scheduler
QoS Control
I/O malleability decision
Startup
WP2
Ad-hoc Storage
HSM data flow
Back-end storage
job queue, system, application, and I/O states + QoS + user hints
I/O malleability decision
I/O state
Ad-hoc Storage system monitoring
SLURM
job queue state + user hints
job malleability decision
WP6
Intelligent Controller
Distributed Database IC
ADMIRE-enabled applications + user hints
Control commands
System state
ADMIRE application manager
WP7
Applications
Job + I/O malleability decisions
job queue, system, application, and I/O states + user hints
System, storage, application states
Monitoring commands
Monitoring: applications
WP3
Malleability Manager
WP5
Monitoring Manager
Performance Database
WP5
Sensing and Profiling
Monitoring
Monitoring: LIME + Paratools TAU

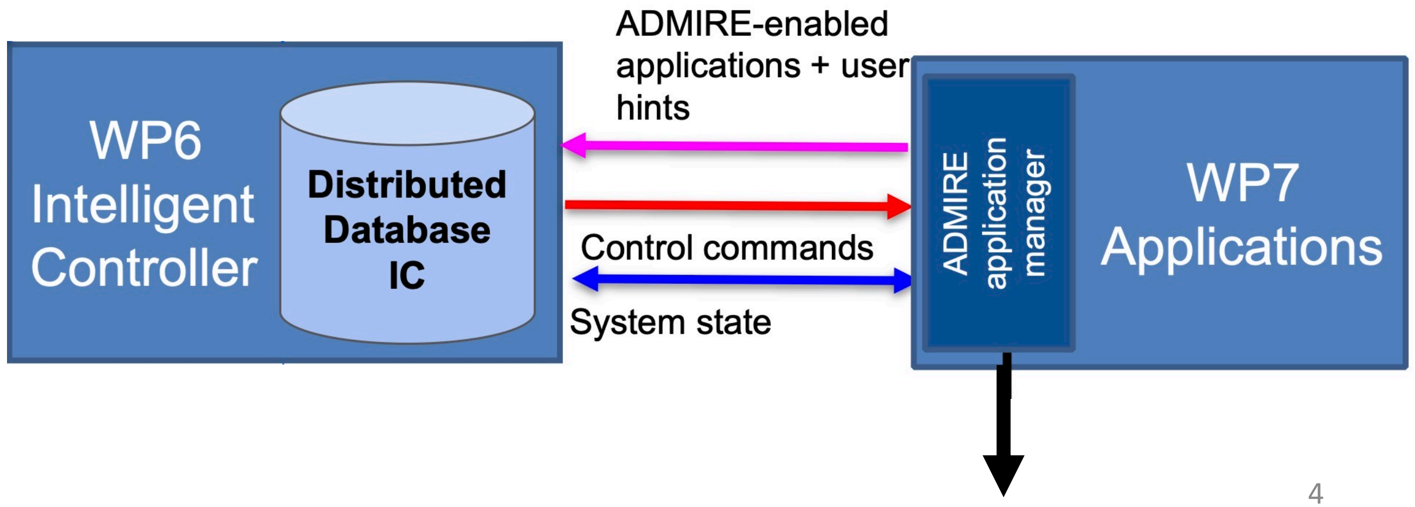
Application malleability



QoS Control
WP4
I/O Scheduler
QoS Control
I/O malleability decision
Startup
WP2
Ad-hoc Storage
HSM data flow
Back-end storage
job queue, system, application, and I/O states + QoS + user hints
I/O malleability decision
I/O state
Ad-hoc Storage system monitoring
SLURM
job queue state + user hints
job malleability decision
WP6
Intelligent
Controller
Distributed Database IC
ADMIRE-enabled applications + user hints
Control commands
System state
ADMIRE application manager
WP7
Applications
Job + I/O malleability decisions
job queue, system, application, and I/O states + user hints
System, storage, application states
Monitoring commands
Monitoring: applications
WP3
Malleability Manager
WP5
Monitoring
Manager
Performance Database
WP5
Sensing and Profiling
Monitoring
Monitoring: LIME + Paratools TAU

ADMIRE application manager

- FlexMPI extension to Fortran
- New interface aligned with ADMIRE project
- New communication library: libicc
  - Connection with the Intelligent Controller
  - Connection with Slurm
- New spawn / shrink operations
- Integration with applications
  - WaComm++
  - Nek5000
  - Numerical kernels
  - Epigraph


WP6 Intelligent Controller
Distributed Database IC
ADMIRE-enabled applications + user hints
Control commands
System state
ADMIRE application manager
WP7 Applications

Outline

- Use of malleable functionality
- Global malleable communicator
- Malleability region
- Monitoring service
- Attributes
  - Register/update key-values
  - Get value size and value
  - Example of attributes

Example: iterative kernel

```
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);
  …
  …              Initialization section
  …
  while (it < itmax) {
    …
    …
    …
    …
    …                Iterative section
    …
    …
    …
    MPI_Reduce(..., MPI_COMM_WORLD);
  }
MPI_Finalize();
}
```

ADMIRE malleable data solutions for HPC

Example: iterative kernel

```
int main(int argc, char *argv[]){
   …
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
   MPI_Comm_size(MPI_COMM_WORLD, &world_size);
   …
   …            Initialization section
   …
   while (it < itmax) {
     …
     …
     …
     …
     …             Iterative section
     …
     …
     …
     MPI_Reduce(..., MPI_COMM_WORLD);
   }
MPI_Finalize();
}
```

```
#include <empi.h>
int main(int argc, char *argv[]){
   …
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
   MPI_Comm_size(ADM_COMM_WORLD, &world_size);
   …
   ADM_GetSysAttributes (...)
   …
   while (it < itmax) {
     …
     ADM_MalleableRegion (ADM_START);
     ADM_MonitoringService (ADM_START);
     // Iterative section
     ADM_MonitoringService (ADM_STOP);
     status = ADM_MalleableRegion (ADM_STOP);
     if (status == ADM_REMOVED) break;
     …
     MPI_Reduce(..., ADM_COMM_WORLD);
   }
MPI_Finalize();
}
```

EuroHPC Joint Undertaking

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)`

Example: Malleability Skeleton

```
#include <empi.h>
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
    // Iterative section
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    …
    MPI_Reduce(..., MPI_COMM_WORLD);
  }
MPI_Finalize();
}
```

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 exiting a malleable region.
    - In Fortran this update must be explicit by executing:
      call ADM_GetComm(ADM_COMM_WORLD)

Example: Malleability Skeleton

```
#include <empi.h>
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
     // Iterative section
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    …
    MPI_Reduce(..., ADM_COMM_WORLD);
  }
MPI_Finalize();
}
```

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:
  - Defines the malleability region.
  - At the start of the region hints are sent to the IC and (if necessary) resources are allocated.
  - 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.
  - Registers the Nº of malleability regions executed (`ADM_GLOBAL_ITERATION`)

Example: Malleability Skeleton

```
#include <empi.h>
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
    // Iterative section
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    …
    MPI_Reduce(..., ADM_COMM_WORLD);
  }
MPI_Finalize();
}
```

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 malleability regions.
  - The monitoring happens only within a malleable region.

Example: Malleability Skeleton

```
#include <empi.h>
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
    // Iterative section
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    …
    MPI_Reduce(..., ADM_COMM_WORLD);
  }
MPI_Finalize();
}
```

Attributes: Register/update for simple data

- Register/update integers:
  - **C:** `ADM_RegisterSysAttributesInt("KEY", &intValue);`
  - **F:** `call ADM_RegisterSysAttributesInt("KEY", intValue)`
- Register/update doubles:
  - **C:** `ADM_RegisterSysAttributesDouble("KEY", &realValue);`
  - **F:**
    ```
    call ADM_RegisterSysAttributesDouble("KEY",
                                          realValue)
    ```
- Behaviour:
  - For first time, the new key is register with a copy of the value.
  - If Key already exist:
    - If Value == NULL, key is removed.
    - Else, previous value is replaced by the new one.

Example: Malleability Skeleton

```
#include <empi.h>
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
    // Iterative section
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    …
    MPI_Reduce(..., ADM_COMM_WORLD);
  }
MPI_Finalize();
}
```

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:
  - For first time, the new key is register with a copy of the value.
  - If Key already exist:
    - If Value == NULL, key is removed.
    - Else, previous value is replaced by 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:
  - For first time, the new key is register with a copy of the value.
  - If Key already exist:
    - If Value == NULL, key is removed.
    - Else, previous value is replaced by the new one.

Attributes: Get value for simple data

- 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 already created.
  - 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.

ADMIRE
malleable data solutions for HPC

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.

EuroHPC
Joint Undertaking

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).

Example: Malleability Skeleton

```
int main(int argc, char *argv[]){
  …
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
  MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  …
  ADM_GetSysAttributes ("ADM_GLOBAL_ITERATION", (void *)&it, sizeof(int));
  …
  while (it < itmax) {
    …
    ADM_MalleableRegion (ADM_START);
    ADM_MonitoringService (ADM_START);
    …
    ADM_MonitoringService (ADM_STOP);
    status = ADM_MalleableRegion (ADM_STOP);
    if (status == ADM_REMOVED) break;
    MPI_Comm_rank(ADM_COMM_WORLD, &world_rank);
    MPI_Comm_size(ADM_COMM_WORLD, &world_size);
  }
MPI_Finalize();
}
```

malleable data solutions for HPC

Adaptive multi-tier intelligent data manager for Exascale

ADMIRE interface for malleability

Universidad Carlos III de Madrid

December 12th 2023

Grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1