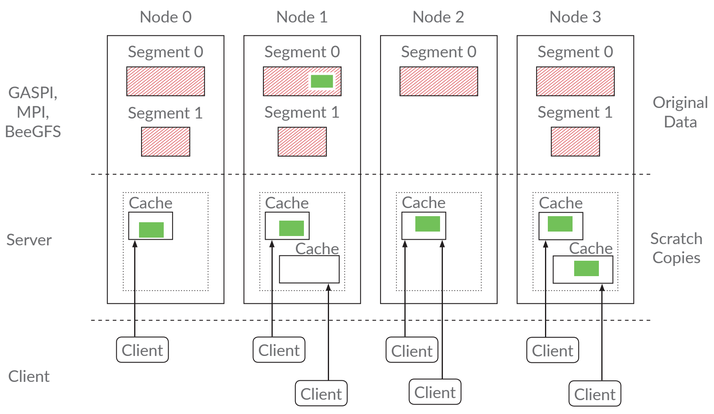
INTERTWinE: Programming Model INTERoperability ToWards Exascale

INTERTWinE addresses the problem of **programming model design and implementation for the Exascale**. The first Exascale computers will be very highly parallel systems, consisting of a hierarchy of architectural levels. To program such systems effectively and portably, programming APIs with efficient and robust implementations must be ready in the appropriate timescale. A single, "silver bullet" API which addresses all the architectural levels does not exist and seems very unlikely to emerge soon enough. We must therefore expect that using **combinations of different APIs at different system levels** will be the only practical solution in the short to medium term. Although there remains room for improvement in individual programming models and their implementations, the main challenges lie in **interoperability between APIs both at the specification level and at the implementation level**. In addition to interoperability directed at specific API combinations, INTERTWinE tackles interoperability on a more general level with the help of the directory cache service and the resource manager.

Directory Cache



Directory Cache architecture

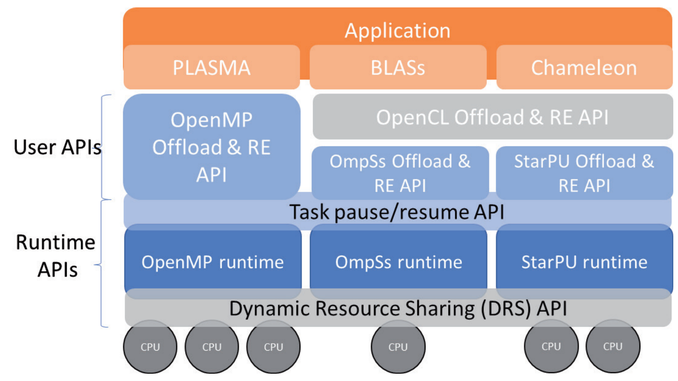
The envisaged directory/cache allows the task-based runtime systems to efficiently run distributed applications on top, while being able to consistently **manage data stored in distributed memory** or in local caches:

- The runtimes (OmpSs, StarPU) are completely independent from the physical representation of data and from the type of storage used
- Access of data through the same interface to an extendable list of memory segment implementations based on different communication libraries (GASPI, MPI)

For applications based on runtimes a **switch to a fully distributed version is almost transparent** with the help of the directory cache.

Targeted API combinations: MPI / GASPI + OmpSs / StarPU (scheduling component of task-based runtimes important for a performant execution with the directory cache)

Resource Manager



Resource Manager APIs: main components and general overview

The main goal of the INTERTWinE Resource Manager is to **coordinate access to CPUs and GPUs resources between different runtime systems and APIs** to avoid both oversubscription and undersubscription situations. We are developing three different APIs:

- An offloading API -based on OpenCL- to call parallel kernels (e.g. a piece of code written in OpenMP or StarPU) from one runtime system to another one
- A dynamic resource sharing API - also based on OpenCL- to indicate what specific CPUs can be used to execute a parallel kernel
- A task pause/resume API so different runtimes in the same process can lend and borrow CPUs between them when they are not using all the resource or they need more resources respectively.

Targeted API combinations: OpenMP / OmpSs / StarPU / mathematical libraries

Parallel Programming Models

MPI

Interoperability MPI plus threads
- Interaction with all non-MPI components other than POSIX-like threads implementation-dependent
- Need to strike a balance between thread safety vs. performance optimization
- Solution: MPI endpoints which introduce a new communicator creation function that creates a communicator with multiple ranks for each MPI process in a parent communicator

INTERTWinE ambition
- MPI endpoints proposal under discussion in MPI Forum with active contribution from INTERTWinE

OpenMP

Description
- Parallel application program interface targeting Symmetric Multiprocessor systems
- Including accelerator devices like GPUs, DSPs or MICs architectures

Interoperability OpenMP plus math. Libs (like Intel MKL)
- Proper use of the underlying computational resources important

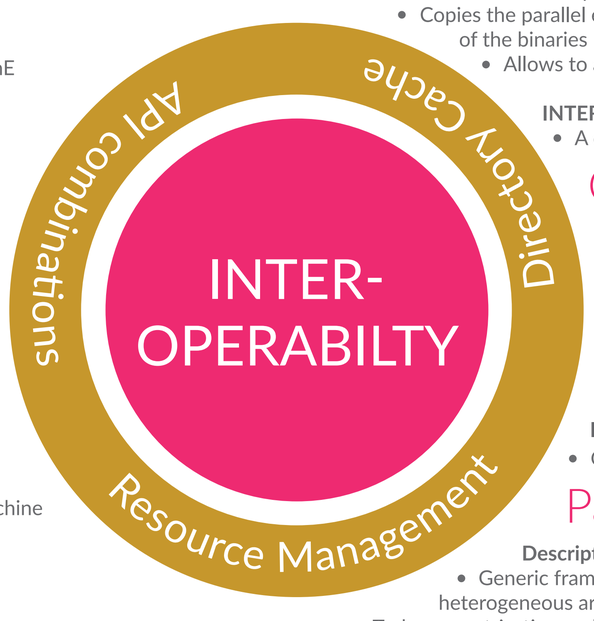
INTERTWinE ambition
- Use of the Resource Manager highly recommendable which will be presented to the OpenMP community

StarPU

Description
- Runtime system which enables programmers to exploit CPUs and accelerator units available on a machine

Interoperability StarPU and MPI
- Supports serving data dependencies over MPI on distributed sessions
- Each participating process annotates data with node ownership
- Each processs submits the same sequence of tasks
- Each task is by default executed on the node where it accesses data in ‚write' mode

INTERTWinE ambition
- Test strategies enabling fully multithreaded incoming messages processing, such as ‚endpoints' (MPI) or ‚notifications' (GASPI)
- Interface with the INTERTWinE resource manager and directory cache service



GASPI

Description
- Defines asynchronous, single sided and non blocking communication primitives for a Partitioned Global Address Space (PGAS)

Interoperability GASPI plus MPI
- Allows for incremental porting of existing MPI applications
- Copies the parallel environment during its initialization → keep existing toolchains, including distribution and initialization of the binaries
- Allows to access data that were allocated in the MPI program without additional copy

INTERTWinE ambition
- A closer memory and communication management of GASPI and MPI

OmpSs

Description
- Prog. model exploiting task-based parallelism for applications written in C, C++ or Fortran

Interoperability GASPI plus MPI
- Extended to run in a multi-node cluster environment with a masterslave design: one process acts as director of the execution (master) and the rest of processes (slaves) follow the commands issued by the master

INTERTWinE ambition
- OmpSs will apply the resource manager and directory cache service of INTERTWinE

PaRSEC

Description
- Generic framework for architecture-aware scheduling and management of micro-tasks on distributed many-core heterogeneous architectures
- Task parametrisation and provision of architecture-aware scheduling

Interoperability PaRSEC and MPI
- Coexistence of PaRSEC-based numerical libraries (e.g. DPLASMA) with MPI applications to be investigated

INTERTWinE ambition
- Interface with the INTERTWinE Resource Manager and experiment with the Directory Cache service

Prioritized API combinations

To ensure interoperability between all six programming APIs featured in INTERTWinE (plus possibly also CUDA, OpenCL and MKL) one has to consider at least 15 API combinations. To make such an approach more reasonable certain **API combinations have been prioritized** and will be main goal of interoperability efforts between the APIs and interoperability studies of the Co-design applications. A few combinations like MPI / GASPI plus OmpSs / StarPU **will be tackled with the help of the described directory cache service**. Others like PLASMA plus OmpSs / StarPU, etc. **will validate the resource manager functionality**. The main remaining API combinations studied in more detail are: MPI plus OpenMP and GASPI plus MPI.

Co-Design Apps

The goal of the Co-design applications is to **provide a set of applications/kernels and design benchmarks** to permit the exploration of interoperability issues. The applications **evaluate** the enhancements to **programming model APIs** and runtime implementations and **feedback** their experience **to the Resource Manager and Directory Cache service**. Some preliminary studies and first recommendations to the programming model APIs are sketched.

Ludwig

- Description: Simulation of complex fluid mixtures
- Interoperability studied: MPI and GASPI in the halo exchange which is required at each time step of the simulation
- Interoperability issue: MPI data types which are not supported by GASPI -> requires unpacking of MPI data types and then copy operation into a continuous data segment
- Comparision between MPI and GASPI: GASPI performs at the same level of optimized MPI for large messages
- Recommendation: GASPI look into the transparent handling of MPI data types
- INTERTWinE interoperability targets: MPI + GASPI, MPI (w + wo endpoints) + OpenMP (w + wo task-based), MPI+OmpSs / StarPU

BAR

- Type of code: Barcelona Application Repository, set of kernels (Cholesky factorization, matrix multiplication, the heat and N-body benchmark), based on the OmpSs programming model
- Interoperability studied: OmpSs plus OpenMP in the N-body simulation
- Comparision between OmpSs and OmpSs+OpenMP: performance in OmpSs+OpenMP decreased due to missing resource management of the underlying CPU resources
- Recommendation: use the INTERTWinE resource manager for the OmpSs + OpenMP
- INTERTWinE interoperability targets: StarPU / OmpSs + MKL, MPI (w and wo endpoints) + OmpSs, OmpSs + CUDA / OpenCL

(D)PLASMA

- Type of code: PLASMA (aiming at shared memory architectures) and DPLASMA (aiming at distributed memory environments), parallel libraries for numerical linear algebra with dense matrices
- Interoperability studied: conversion of PLASMA from its own runtime system; QUARK, to the OpenMP task parallelism, DPLASMA relies on the PaRSEC runtime system, using MPI message passing internally
- INTERTWinE interoperability targets: maintain smooth interoperability with MPI, OpenMP, OmpSs, StarPU

iPIC3D

- Description: C++ MPI plus OpenMP work sharing particle-in-cell (PIC) application for the simulation of space and fusion plasmas during the interaction between the solar wind and the Earth magnetic field
- Interoperability studied: multithreaded MPI approach with OpenMP tasks
- Comparison: slightly lower performance with OpenMP tasks
- INTERTWinE interoperability targets: MPI + OpenMP (w and wo task-based) GASPI + OpenMP MPI + GASPI

TAU

- Type of code: CFD solver for aeronautics (hybrid unstructured solver for the Navier-Stokes equations), next generation solver multithreaded within single domains, use either MPI or GASPI for network communications
- Interoperability studied: potential of task based programming models like OmpSs and StarPU compared to flat threading model to master the deep and fragmented memory hierarchies of next generation systems
- Recommendation: use GASPI as global extension of OmpSs
- INTERTWinE interoperability targets: MPI / GASPI + OpenMP, MPI / GASPI + OmpSs

Graph-Blas

- Type of code: computation with large-scale graphs (combinatorial computing)
- Interoperability studied: OmpSs+MKL
- Interoperability issues: optimized scale of logical partitions to run with OmpSs,, OmpSs plus multi-threaded MKL oversubscription due to a bad mapping of threads to cores
- Recommendation: leverage the oversubscription by using the INTERTWinE Resource Manager
- INTERTWinE interoperability targets: OpenMP / OmpSs + MKL, MPI + OmpSs / OpenMP

If you are interested in more information, and/or to sign up for the INTERTWinE newsletter : **www.intertwine-project.eu**
and read our TWITTER feeds: @intertwine_eu
The INTERTWinE project is ongoing from Oct. 2015 to Sept. 2018

The project is funded by the European Commission (grant agreement number: 671602).