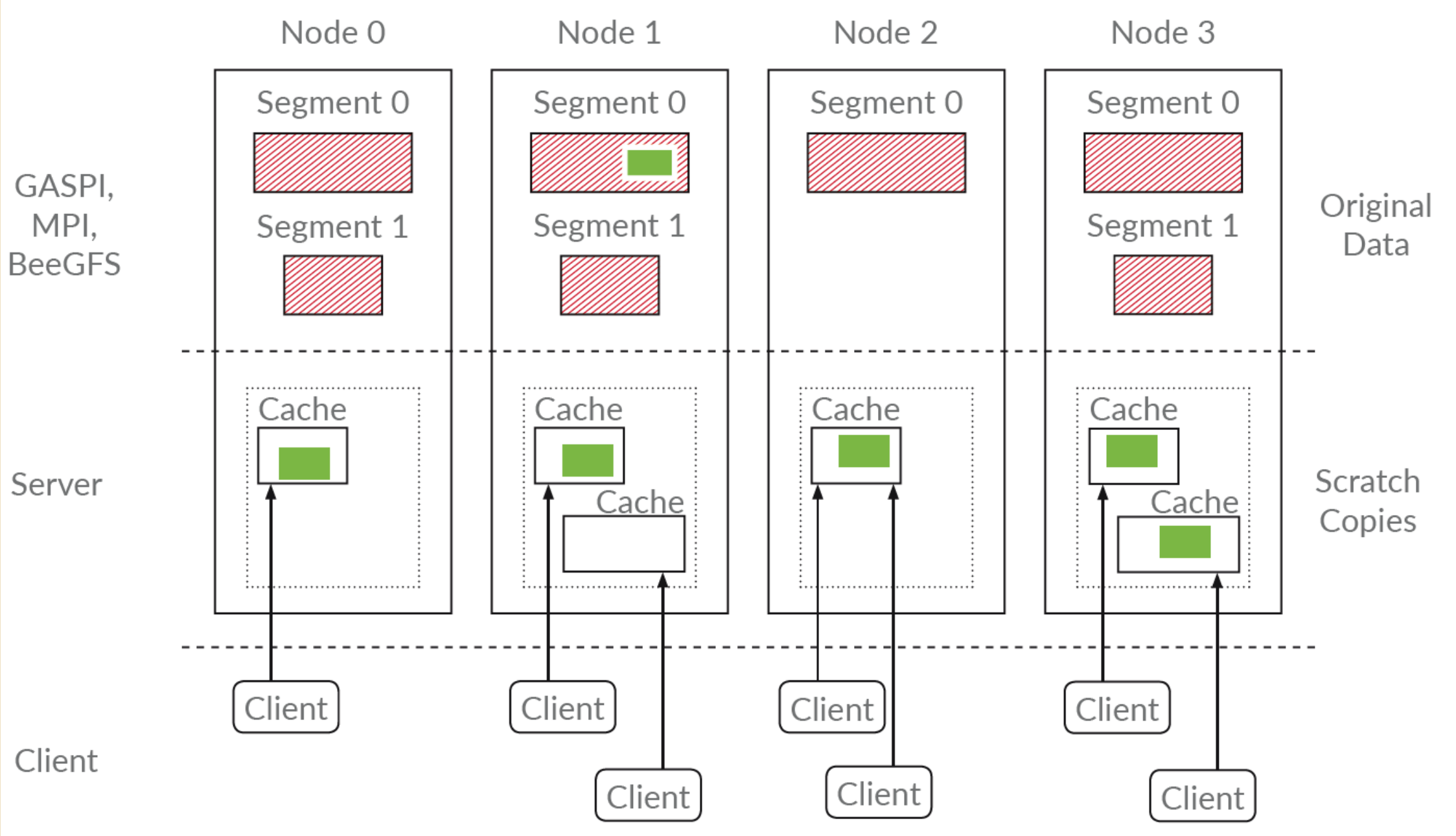
INTERTWinE: Programming Model INTERoperability ToWards Exascale

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 at the specification and implementation levels.** In addition to interoperability among APIs, 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 Directory/Cache API allows the task-based programming models to work with an abstract view of the distributed memory as a single shared address space

The API can be used for consistently managing data stored in distributed memory and for maximizing the cache reuse at node level

Directory/Cache API allows the task-based runtime systems:

- Complete independence from the physical representation of data and from the type of storage used
- Access through the same interface to an extendable list of memory segment implementations (GASPI, MPI, BeeGFS, etc.)
- To benefit from automatic caching

The Directory/ Cache enables the interoperability between task-based models and distributed memory technologies

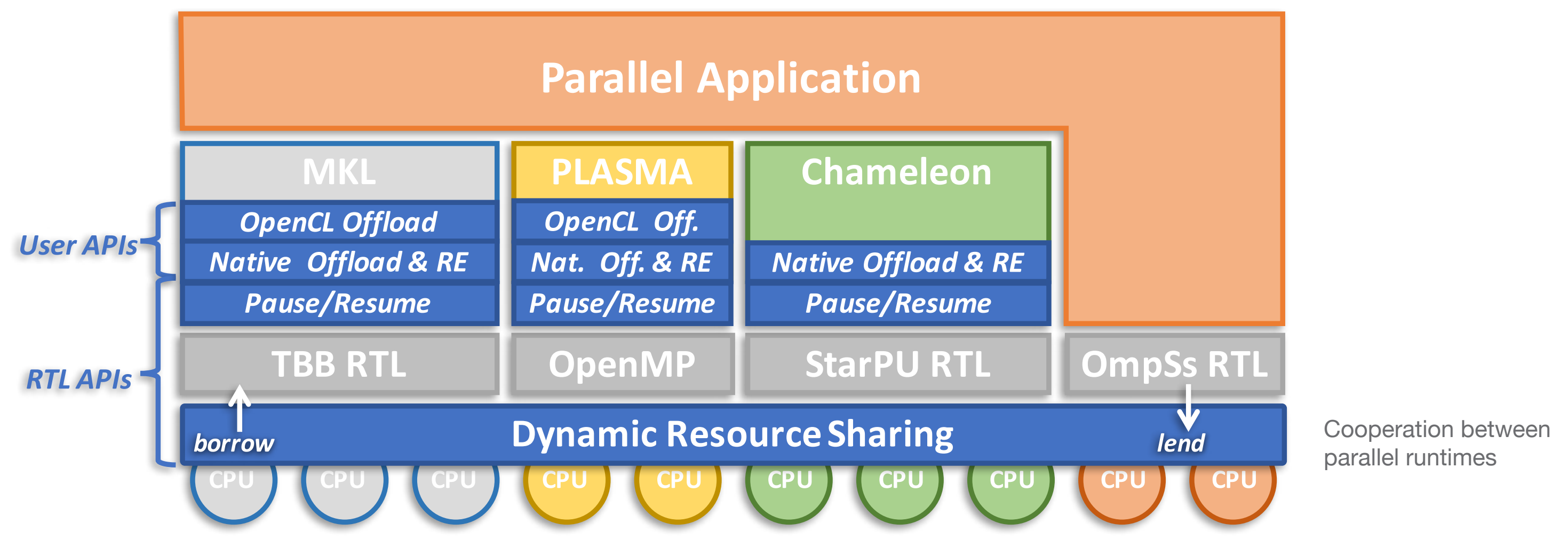
A prototype implementation of the Directory/ Cache API is already available and is currently tested by several runtime systems (OmpSs, GPI-Space, PaRSEC, StarPU)

Resource Manager

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

- An offloading API to invoke parallel kernels (e.g. a piece of code written in OpenMP, OmpSs, or StarPU) into a specific set of CPUs from one runtime system to another one
- A dynamic resource sharing API to transparently lend and borrow CPUs between parallel runtimes to avoid under-utilization scenarios
- A task pause/ resume API to improve the interoperability of task-based APIs with blocking message-passing APIs such as MPI.

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



Parallel Programming Models

MPI

Description

- Message passing API, widely used for distributed memory parallel programming

Interoperability of MPI plus threads

- Interaction with all non-MPI components other than POSIX-like threads is implementation-dependent
- Need to strike a balance between thread safety vs. performance optimization
- Solutions: MPI endpoints, which creates a communicator with multiple ranks for each MPI process; and MPI finepoints, which allows multiple threads to contribute message data to send operations

INTERTWinE ambition

- MPI endpoints and finepoints proposals under discussion in MPI Forum with active contribution from INTERTWinE

OpenMP

Description

- Parallel application program interface targeting Symmetric Multiprocessor systems

Interoperability of OpenMP and other task-based programming models such as StarPU and OmpSs

- Avoid oversubscription and undersubscription scenarios

INTERTWinE ambition

- Use of the Resource Manager APIs to coordinate access to shared CPU resources

StarPU

Description

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

Interoperability of StarPU and MPI

- Supports serving data dependencies over MPI on distributed sessions
- Each participating process annotates data with node ownership
- Each process 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 multi-threaded incoming messages processing, such as ‘endpoints’ (MPI) or ‘notifications’ (GASPI)
- Interface with the INTERTWinE Resource Manager and the Directory/Cache service

GASPI

Description

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

Interoperability of GASPI plus MPI

- Allows for incremental porting of existing MPI applications
- Copies the parallel environment during its initialization, so keeping 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. The concept of shared memory MPI windows is extended with shared GASPI notifications, such that all ranks with access to the window can leverage shared GASPI weak synchronization primitives. We expect that this feature will significantly advance the migration of legacy code towards an asynchronous, data-flow driven communication model.

OmpSs

Description

- Programming model exploiting data-flow parallelism for applications written in C, C++, or FORTRAN

Interoperability with message passing libraries such as MPI and GASPI

- Improve the performance and programmability of hybrid MPI/ GASPI + OmpSs applications by coordinating task scheduling with the message passing progress engine

INTERTWinE ambition

- OmpSs will be extended to work better with MPI and GASPI communication primitives
- Interface with the INTERTWinE Resource Manager and the Directory/Cache service

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 of PaRSEC and OpenMP

- Combination of PaRSEC-based codes (e.g. DPLASMA) with OpenMP applications (e.g. PLASMA) on nodes

INTERTWinE ambition

- Interface with the INTERTWinE Directory/Cache service

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 the **Directory/Cache service**. Some preliminary studies and first recommendations to the programming model APIs are sketched. Visit our Developer Hub for more details and recent updates: **www.intertwine-project.eu/developer-hub**

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 team pursue the transparent handling of MPI data types
- INTERTWinE interoperability targets:

MPI plus GASPI, MPI (w/ and w/o endpoints) plus OpenMP Threads/ Tasks, MPI plus 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 plus OpenMP: performance in OmpSs plus OpenMP decreased due to missing resource management of the underlying CPU resources
- Recommendati on: use the INTERTWinE Resource Manager for the OmpSs plus OpenMP
- INTERTWinE interoperability targets:

StarPU/ OmpSs plus MKL, MPI (w/ and w/o endpoints) and OmpSs, OmpSs and CUDA/ OpenCL.

(D)PLASMA

- Type of code: PLASMA (aiming at shared memory architectures) and DPLASMA (aiming at distributed memory environments) are parallel libraries for numerical linear algebra with dense matrices
- Interoperability studied: taken part in converting PLASMA from its own runtime system (QUARK) to the OpenMP task parallelism
- Parts of PLASMA converted to OmpSs and StarPU runtimes to enable using the Resource Manager
- DPLASMA relies on the PaRSEC runtime system, using MPI message passing internally
- INTERTWinE interoperability targets: maintain smooth interoperability with 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. Currently, the code is presented in three programming models: with multi-threaded MPI enabled, with added OpenMP tasking on top of it, and with the GASPI halo exchange communication
- Comparison: the version with just multi-threaded MPI enabled shows the best performance, while adding OpenMP tasking gives a lower performance due to the OpenMP task creation runtime overhead; the GASPI version showed the promising performance results
- INTERTWinE interoperability targets: MPI plus OmpSs and MPI plus GASPI

TAU

The CFD solver for aeronautics is a hybrid unstructured solver for the Navier-Stokes equations

Next-generation implicit methods are investigated for a new flow solver that works multithreaded within single domains and can use either MPI or GASPI for the network communication. INTERTWinE’s ambition is to evaluate node-local scalability of implicit methods and the potential of using task-based programming models like OmpSs or StarPU. It is anticipated that a task-based approach can be beneficial for upcoming, next-generation systems with deep and fragmented memory hierarchies. Due to its focus on asynchronous one-side dataflow notifications the GASPI API will be an excellent match for this anticipated global extension of OmpSs

Graph-Blas

- Description: Computation with large-scale graphs (combinatorial computing) is crucial for Big Data analytics. While graph computations are often a source of poorly scalable parallel algorithms, due to their irregular nature and low computational intensity, many graph operations exhibit ample coarse-grained parallelism, which can be uncovered by exploiting the duality between graphs and sparse matrices
- Interoperability studied: Combining OmpSs with Intel MKL and MPI with OmpSs in the Graph-Blas algorithms
- OmpSs plus Intel MKL: INTERTWinE avoids oversubscription thanks to the Resource Manager
- MPI plus OmpSs: The Graph-BLAS application exploits the new features developed as part of the INTERTWinE project that improve taskification„ of MPI communication. This optimization allows to overlap computation and communication, improving the interoperability between MPI plus OmpSs and reducing the execution time of graph applications
- INTERTWinE interoperability targets: OmpSs plus MKL and MPI plus 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 project is funded by the European Commission (grant agreement number: 671602).