INTERTWinE – Programming Model
Interoperability towards Exascale

Highlights of the project
INTERTWinE is all about interoperability: making sure that tried and tested ways to program a supercomputer work together effectively and efficiently.

- Extreme-scale parallelism exposed in multiple layers of hardware will require the use of multiple APIs in a single program to deliver Exascale performance.
- There exist interoperability problems when using today’s leading parallel APIs both across different layers, and also within the same layer, which will require improvements to API specification and runtime implementation.
- INTERTWinE is addressing interoperability issues for six key APIs: MPI, GASPI, OpenMP, OmpSs, StarPU and PaRSEC
- Working closely with standards bodies and runtime development teams.

Technology suggested for inclusion in an EsD project

- GPI-2 implementation of GASPI standard, with new interoperability features with MPI (mature, robust). [http://www.gpi-site.com/gpi2/]
- Benchmark suite: kernels and applications ported to a variety of API combinations (in preparation).
- Resource Manager: APIs and reference implementation for effective resource sharing between multiple runtimes on the same node (prototype under development). [http://www.intertwine-project.eu/about-intertwine/resource-manager]
- Directory/Cache: API and reference implementation supporting task scheduling across distributed systems on top of different communication transport layers (prototype, under development). [http://www.intertwine-project.eu/about-intertwine/directory-cache]
- Integrated MPI and OmpSs runtimes supporting efficient message passing within tasks (in development).
- StarPU runtime with enhanced support for distributed memory systems (in development).
- PaRSEC runtime with enhanced interoperability features supporting PLASMA and DPLASMA linear algebra libraries (in development).

How should this technology be used?
All are runtime implementations with user-level APIs; except the benchmark suite, which can be used for validating functionality and testing performance.

Are there any pre- or co-requisite items?
Some normal dependencies on Linux environment, compilers and network interfaces.

Extra work/interaction needed to make them ready?
- Additional effort to make Resource Manager and Directory/Cache implementations production-ready.
- Possible effort required to port runtimes to exotic OS or H/W environments.

Information / actions are needed to best prepare for EsD projects?
Details of HW and OS environments for proposed EsD systems and any key demonstrator applications.

www.intertwine-project.eu