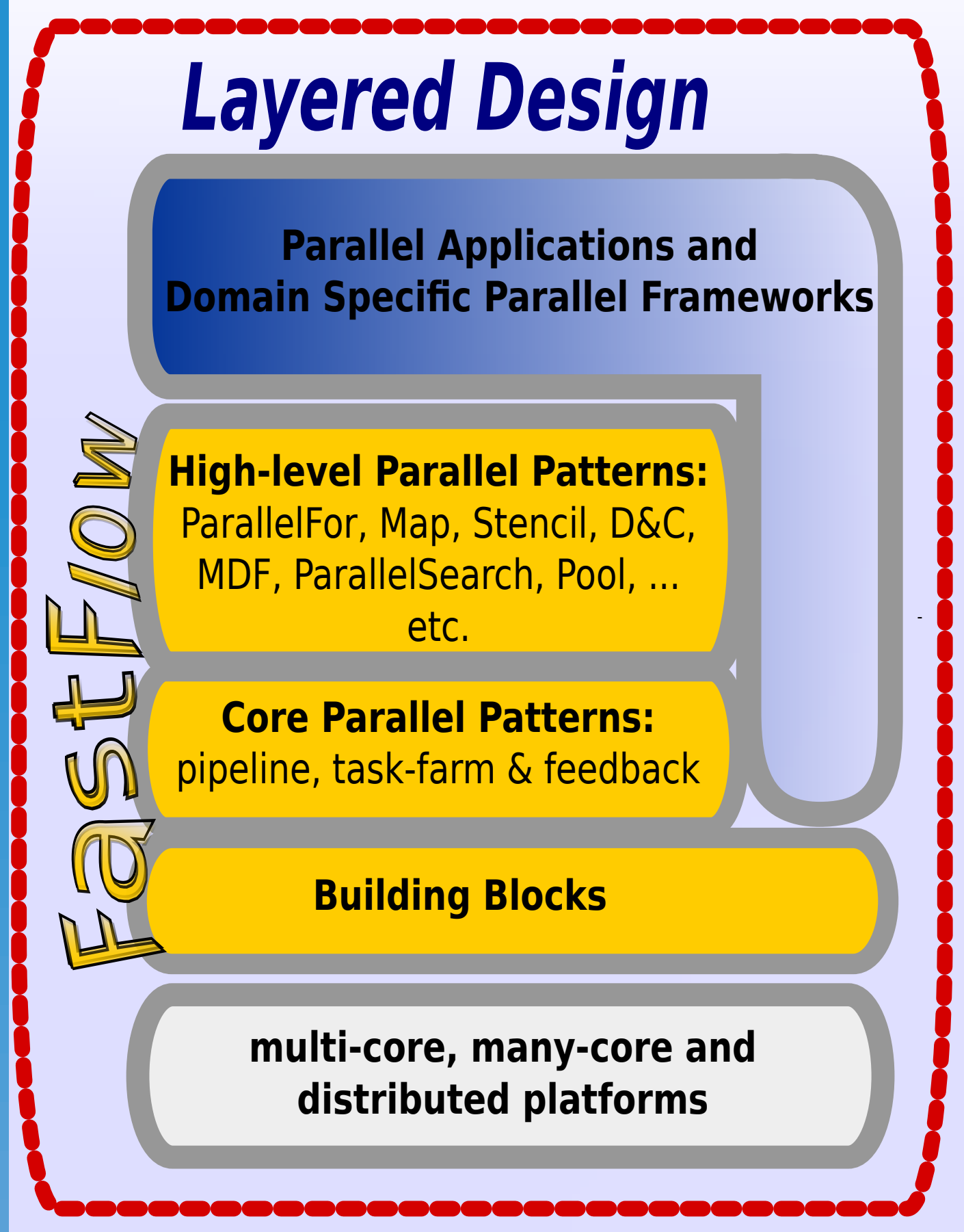
The FastFlow programming framework

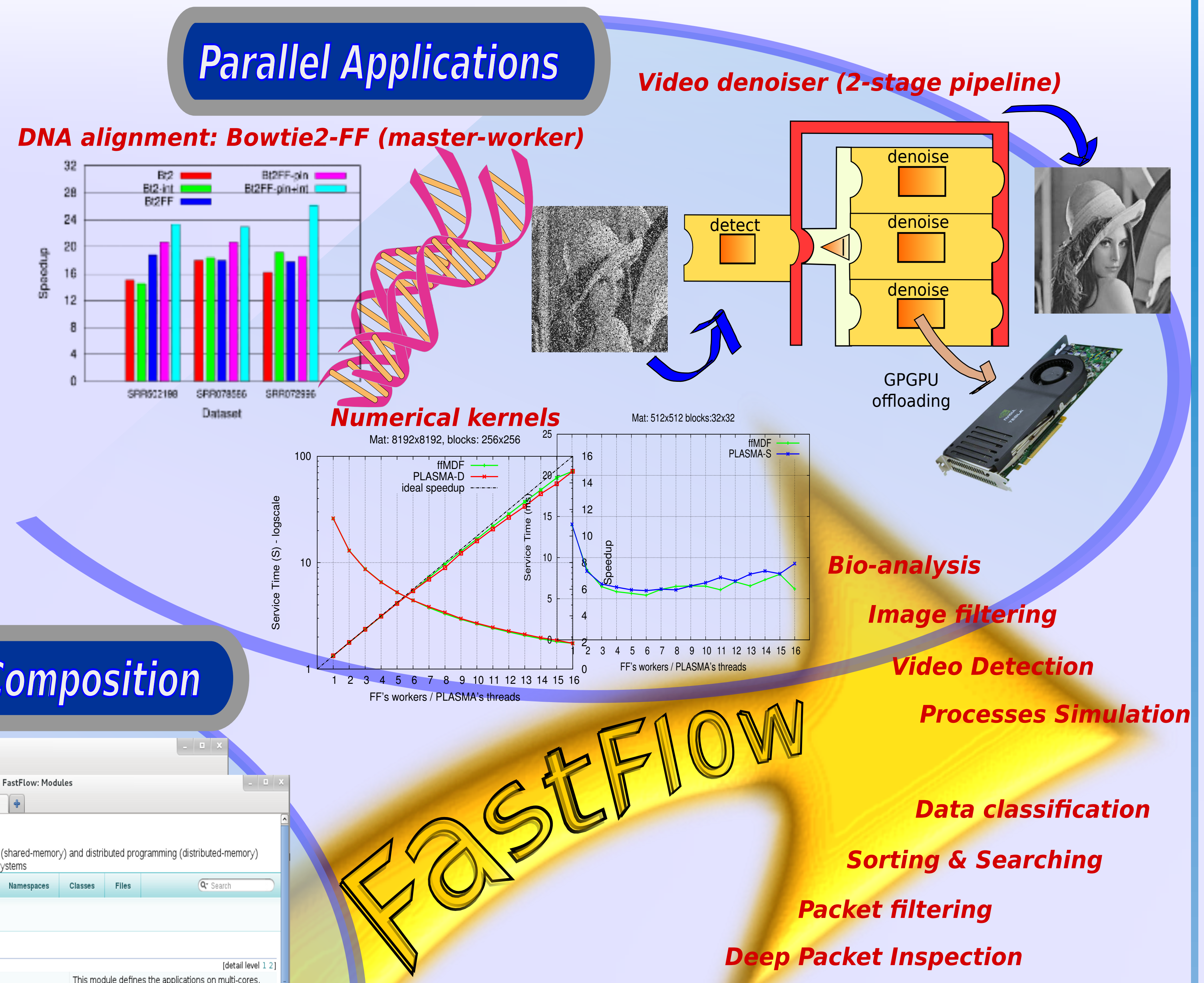
Marco Danelutto° Massimo Torquati° and Marco Aldinucci†
° University of Pisa, Italy † University of Torino, Italy

What

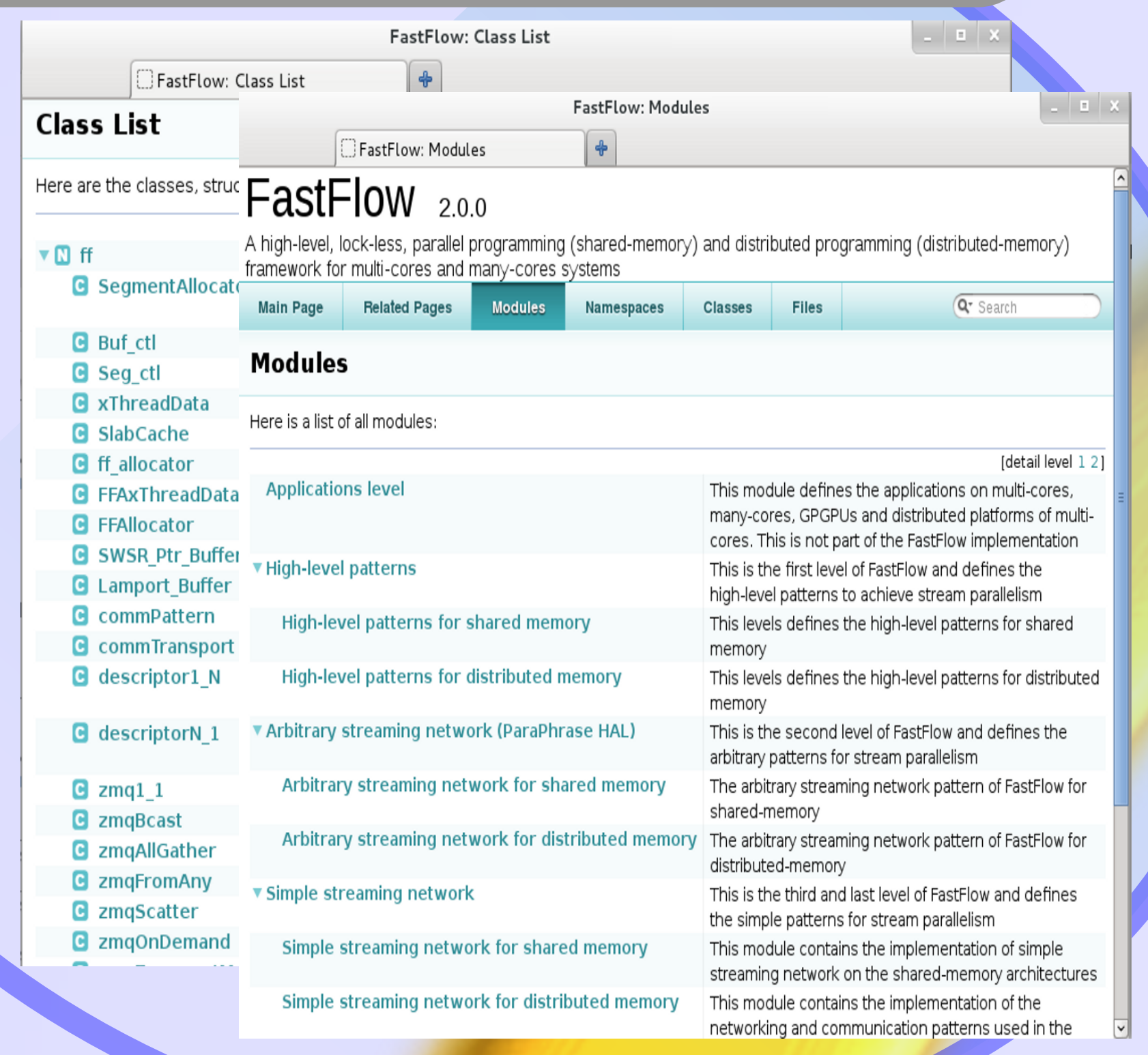
FastFlow is a *parallel programming framework for multi and many core platforms* based upon non-blocking synchronization mechanisms. The framework is structured as a *stack of layers* that provide different levels of abstraction to the application programmer. FastFlow provides the parallel programmer with a set of ready-to-use, parametric algorithmic skeletons modeling the most common parallelism exploitation patterns.

How, the FastFlow way


Layered Design
Parallel Applications and Domain Specific Parallel Frameworks
High-level Parallel Patterns: ParallelFor, Map, Stencil, D&C, MDF, ParallelSearch, Pool, ... etc.
Core Parallel Patterns: pipeline, task-farm & feedback
Building Blocks
multi-core, many-core and distributed platforms
FastFlow



Pattern Selection & Composition



FastFlow: Class List

Class List

Here are the classes, struc

ff
SegmentAllocat
Buf_ctl
Seg_ctl
xThreadData
SlabCache
ff_allocator
FFAxThreadData
FFAllocator
SWSR_Ptr_Buffe
Lamport_Buffer
commPattern
commTransport
descriptor1_N
descriptorN_1
zmq1_1
zmqBcast
zmqAllGather
zmqFromAny
zmqScatter
zmqOnDemand

FastFlow: Modules

FastFlow 2.0.0

A high-level, lock-less, parallel programming (shared-memory) and distributed programming (distributed-memory) framework for multi-cores and many-cores systems

Main Page | Related Pages | Modules | Namespaces | Classes | Files

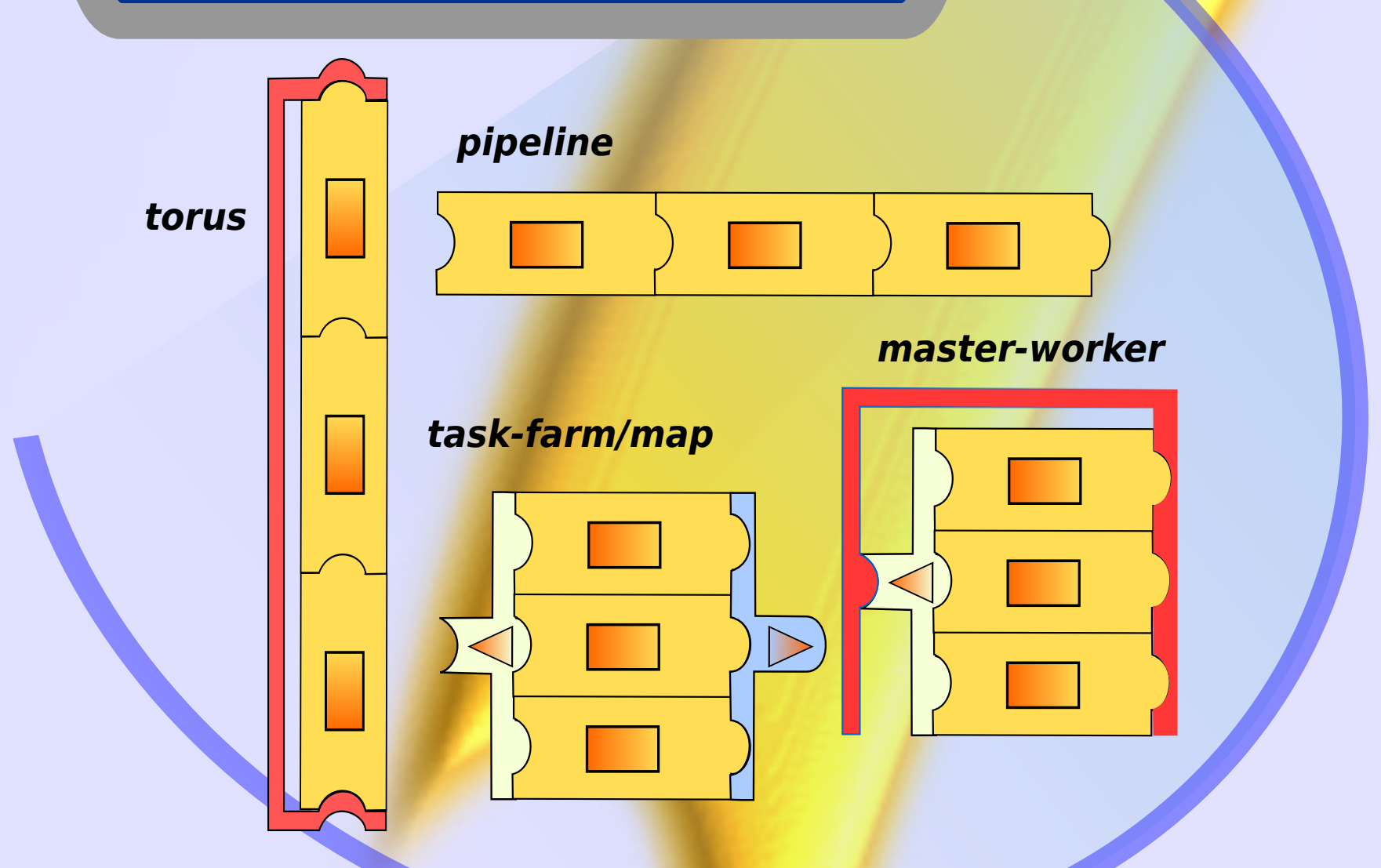
Modules

Here is a list of all modules:

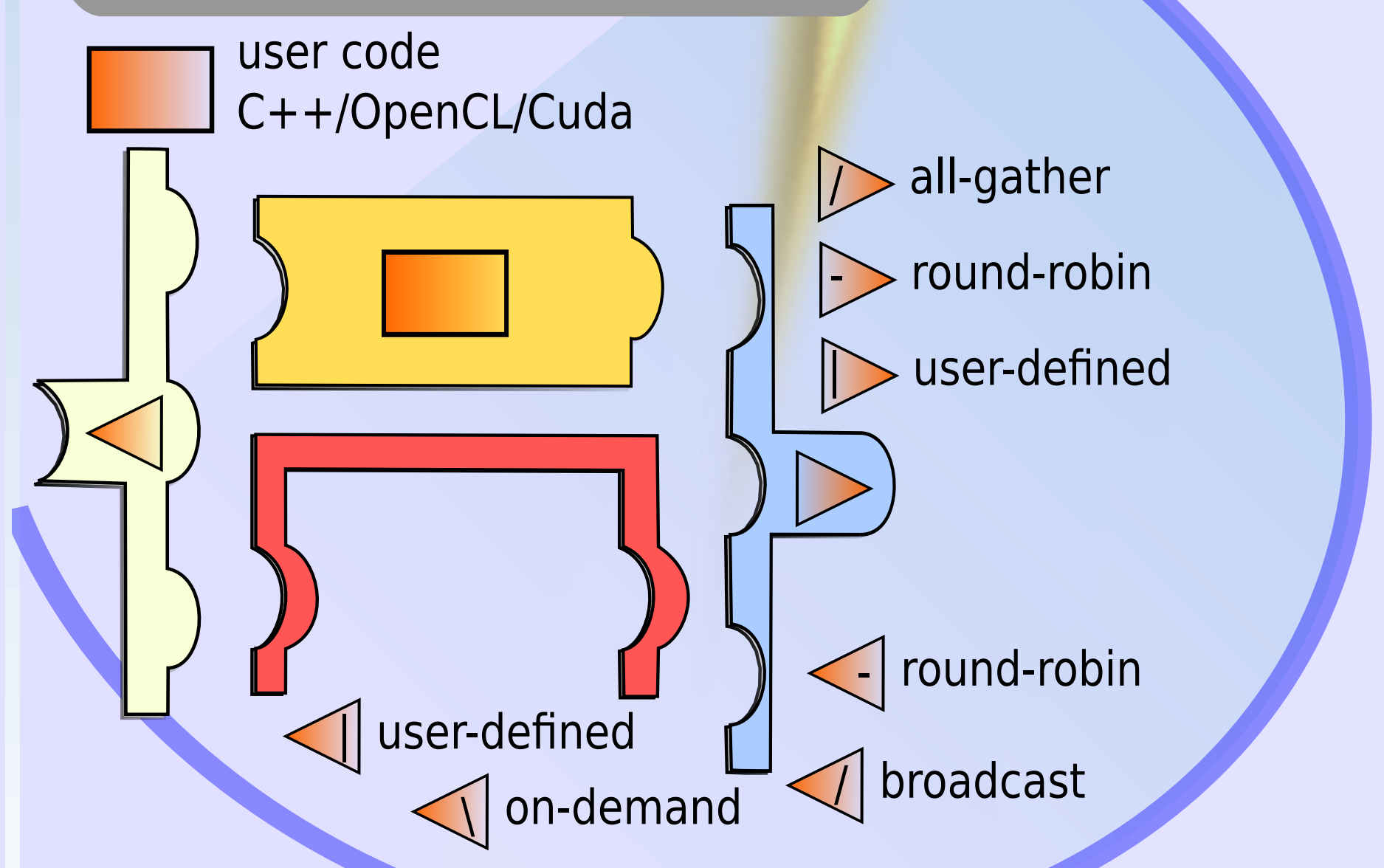
[detail level 1 2]

Applications level	This module defines the applications on multi-cores, many-cores, GPGPUs and distributed platforms of multi-cores. This is not part of the FastFlow implementation
High-level patterns	This is the first level of FastFlow and defines the high-level patterns to achieve stream parallelism
High-level patterns for shared memory	This levels defines the high-level patterns for shared memory
High-level patterns for distributed memory	This levels defines the high-level patterns for distributed memory
Arbitrary streaming network (ParaPhrase HAL)	This is the second level of FastFlow and defines the arbitrary patterns for stream parallelism
Arbitrary streaming network for shared memory	The arbitrary streaming network pattern of FastFlow for shared-memory
Arbitrary streaming network for distributed memory	The arbitrary streaming network pattern of FastFlow for distributed-memory
Simple streaming network	This is the third and last level of FastFlow and defines the simple patterns for stream parallelism
Simple streaming network for shared memory	This module contains the implementation of simple streaming network on the shared-memory architectures
Simple streaming network for distributed memory	This module contains the implementation of the networking and communication patterns used in the

Parallel Patterns



Building Blocks

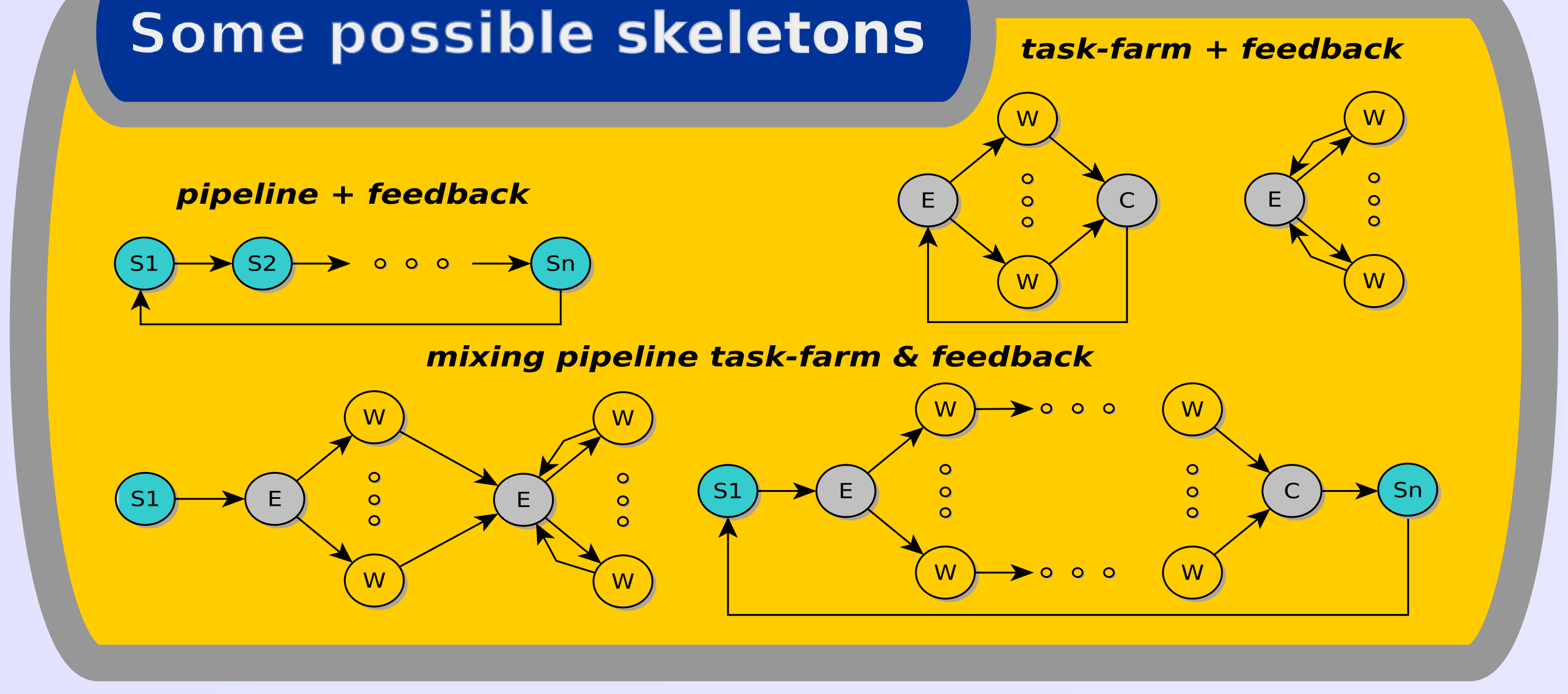


Separation of concerns

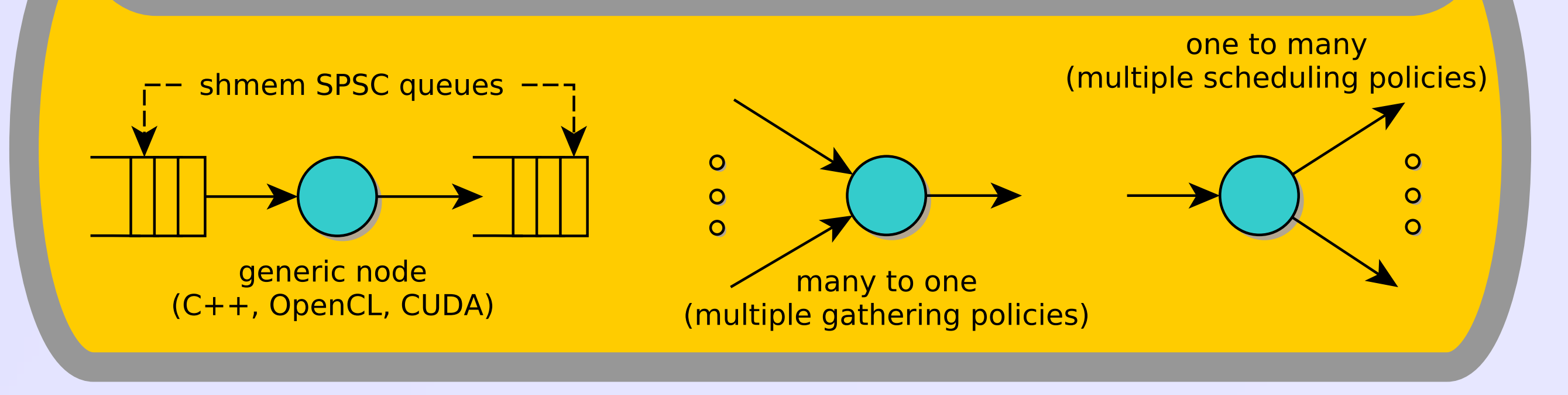
Application programmer :
the application programmer is in charge of **selecting the proper pattern** from the ones provided by the framework and connecting them in order to create a suitable streaming network able to solve the problem at hand.

System programmer :
the system programmer either uses the low-level building blocks to create **new patterns** or uses and optimizes the composition of the existing patterns to build new skeletons for the target platform.

Some possible skeletons



Efficient Building Block Implementation


shmem SPSC queues
generic node (C++, OpenCL, CUDA)
many to one (multiple gathering policies)
one to many (multiple scheduling policies)

Targeting CPUs+GPGPUs

Mixing the computation on both CPUs and GPGPUs is supported in FastFlow via both OpenCL and CUDA guest code. The OpenCL-based *map* parallel pattern may execute tasks on one or more GPGPUs and in parallel with the available CPUs. Building complex streaming networks having different parts of the net running on GPGPUs and part on CPUs is straightforward and effective.

Distributed Systems

FastFlow supports heterogeneous distributed platforms. For TCP/IP networks, the ZeroMQ library (www.zeromq.org) is used while for Infiniband networks a native RDMA-based library has been implemented. Porting FastFlow applications on a distributed environment is straightforward!

Flexible Layered Design

Fastflow is a C++ class library designed as a stack of three main layers. The different layers have two main purposes: 1) *promoting high-level parallel programming*, i.e. explicit pattern-based parallel design of applications, and 2) *to be flexible and efficient* for programming multi and many core platforms. The FastFlow programmer may choose to use the mechanisms provided by the layers which best suit his/her needs.

Building Blocks: This is the lowest level layer. It comprises: i) the *wrapper* nodes (the ff_node derived classes) used to embed existing portions of code (C/C++, OpenCL, CUDA) into parallel programs; and ii) the one-to-many, many-to-one and feedback *combinators* for connecting nodes and routing data in different ways. At this level any asynchronous streaming network can be built; the semantics is data-flow. Each wrapper building block has an input and an output lock-free bounded or unbounded SPSC queue, and their well-defined semantics promotes the possibility to automatically refactor their compositions to better exploit target architecture features.

Core Parallel Patterns: On top of the Building Block layer, has been implemented the *pipeline* and several forms of the *task-farm* implementation skeletons. They can be composed and nested in several different ways using also the feedback component for routing back data streams. The *task-farm* is fully customizable both in terms of scheduling and gathering policies. A pipeline of task-farm components, when nested, may be optimized for reducing the number of concurrent threads.

High-Level Parallel Patterns: This is the top level layer. To structure his/her parallel application, the application programmer, uses the patterns available at this level and their compositions through the pipeline and task-farm core patterns. This layer is still in evolution: more patterns will be developed and further optimizations will be applied to the current ones. The high-level patterns currently available are ParallelFor/Map, ParallelReduce, Stencil, ParallelSearch, Macro-Dataflow, D&C, Pool Evolution.

Platforms supported

FastFlow supports Intel, AMD, IMB Power and ARM general purpose multi-core based platforms. Recently it has also been ported on Tilera Tile*Pro*64 and Intel Xeon PHI such that tasks can be offloaded on these accelerators. FastFlow-based code may be compiled with any recent GNU and Intel compilers.

FP7 Projects using FastFlow

SEVENTH FRAMEWORK PROGRAMME

REPARA

ParaPhrase

References

Project Home:
→ http://mc-fastflow.sourceforge.net
→ http://calvados.di.unipi.it/fastflow

SVN repository:
→ https://svn.code.sf.net/p/mc-fastflow/code