Parallel and Distributed Systems / 4

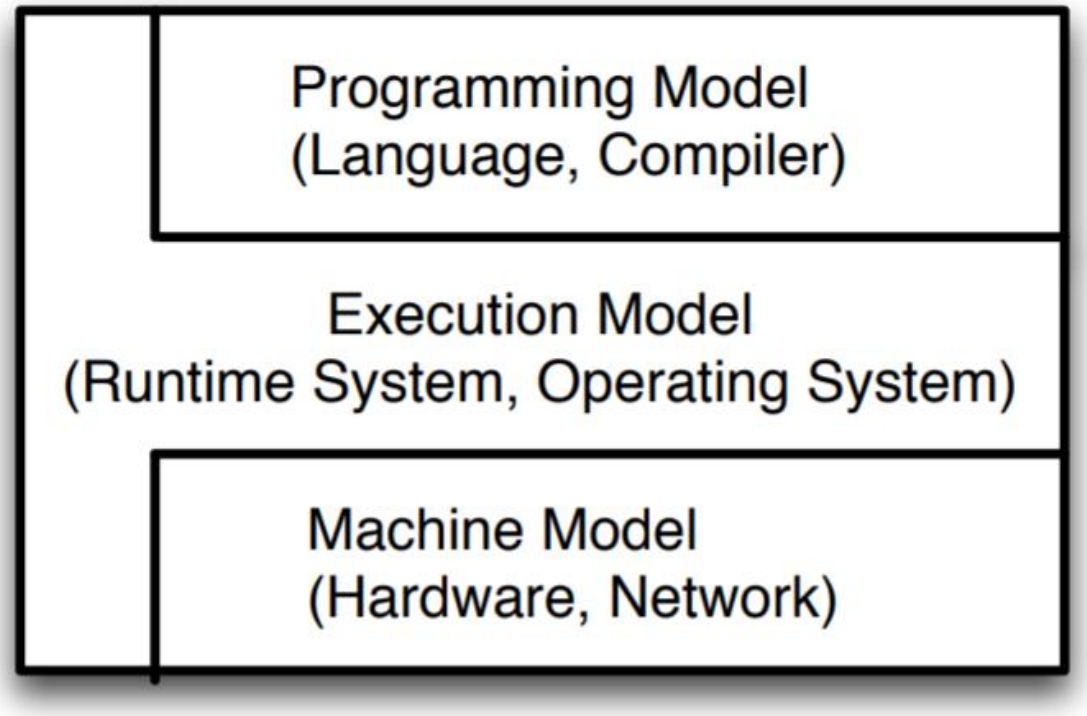
Pavel Krömer,
Dept. of Computer Science,
VSB – Technical University of Ostrava

Agenda

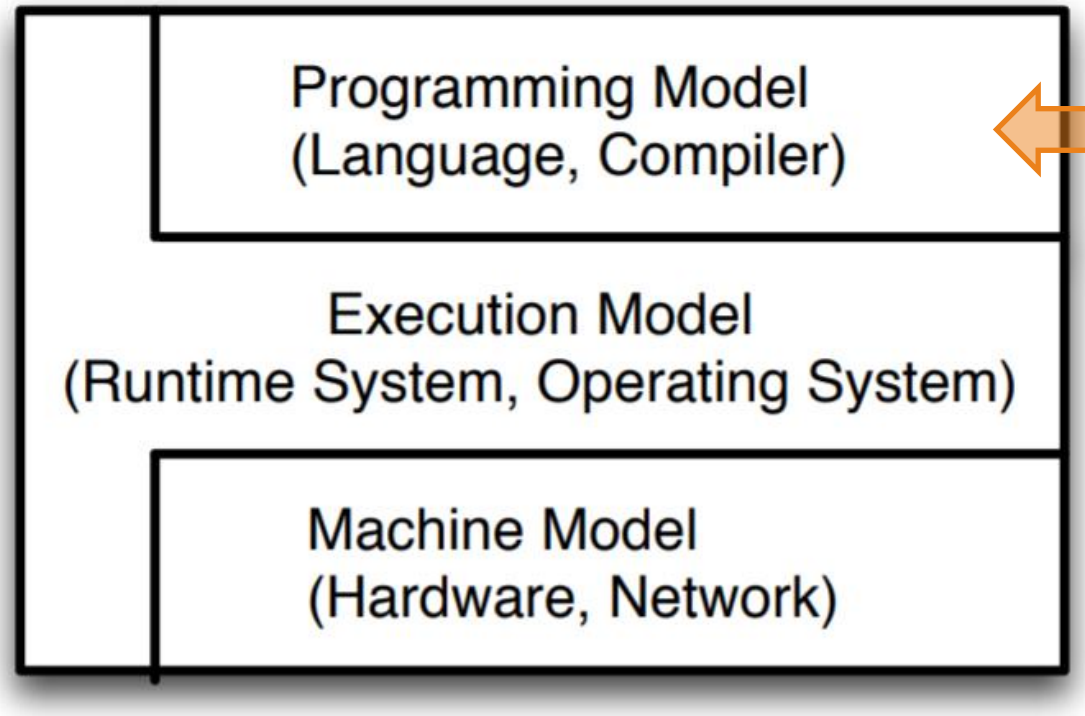
- Models, tools, and frameworks for shared memory multiprocessing
  - already discussed: a dive-in to fork-join with OpenMP

- Literature
  - Michael D. McCool, James Reinders, Arch D. Robison, Structured Parallel Programming: Patterns for Efficient Computation, Elsevier, 2012. Ch. 1: Introduction, pp. 1 – 38
  - Michael Voss, Rafael Asenjo, James Reinders, Pro TBB: C++ Parallel Programming with Threading Building Blocks, Apress, 2019

Components of a parallel programming framework



Components of a parallel programming framework

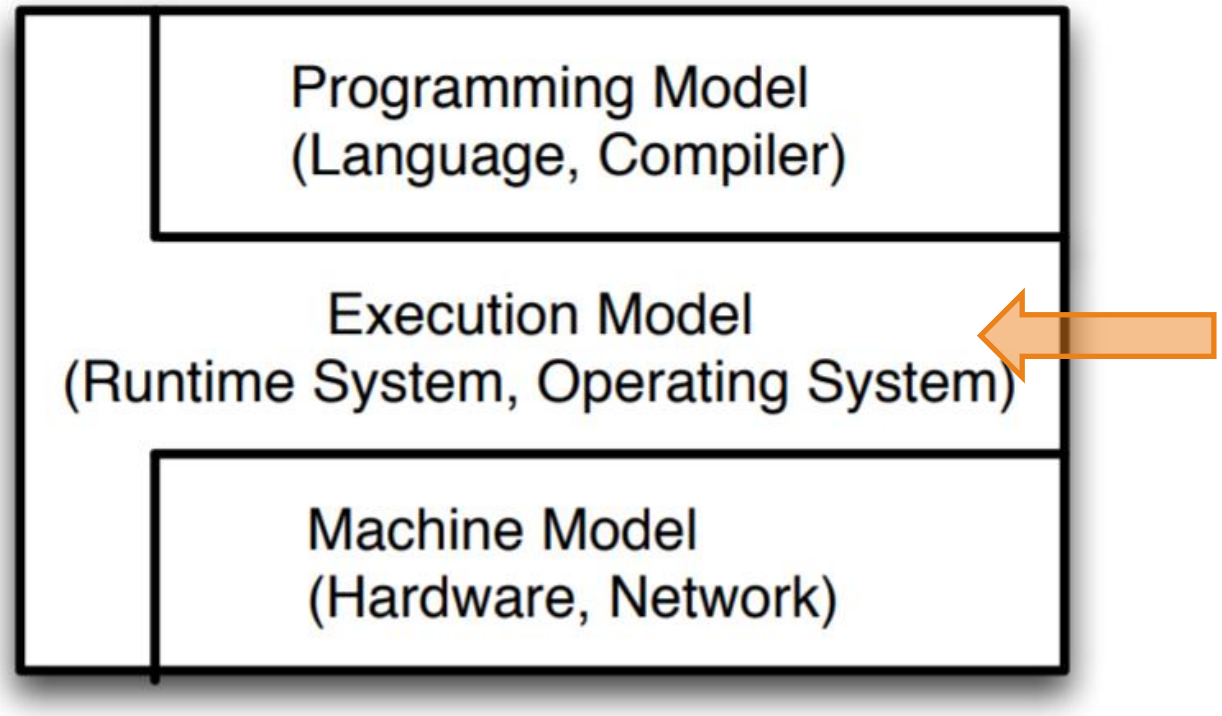


Constructs for exposing and expressing parallelism

- syntax for parallelism in a programming language
- high- and low-level approaches
  - threads vs. parallel sections
  - shared memory vs. message passing

WHAT

Components of a parallel programming framework

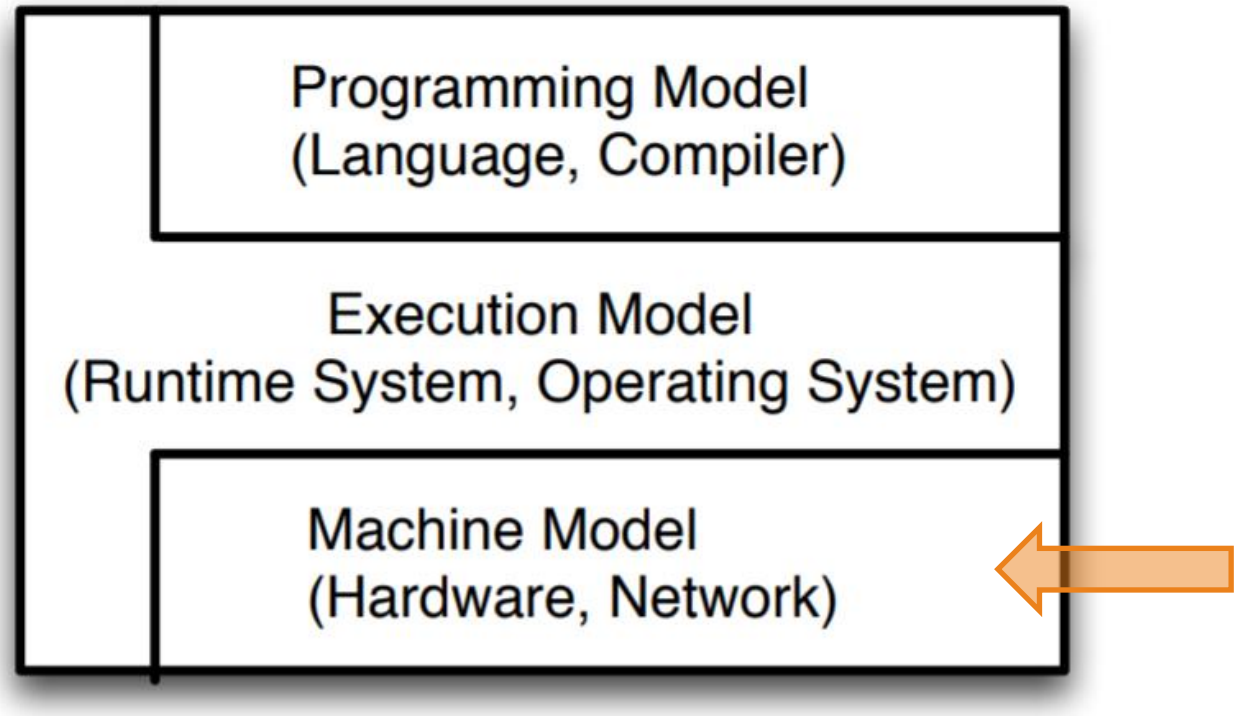


Operational semantics for the model

- an abstraction of computation that links the programming and machine model
- communication, locality, synchronization, etc, used to allow parallelism with maximum efficiency
- yes, the borders are quite fuzzy
- data-parallel; message-driven

HOW

Components of a parallel programming framework



Abstract model isolating the computation from machine details

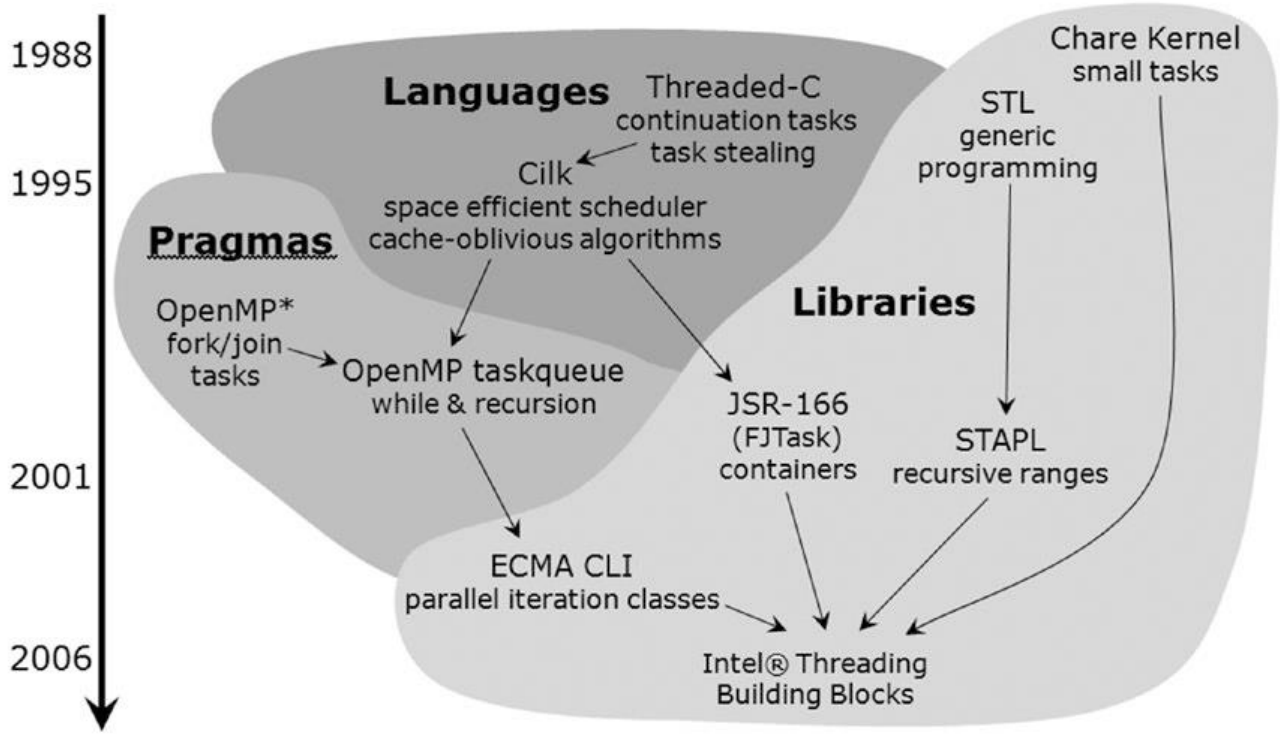
- efficiency, portability, interoperability, tools and infra
- optimization to the level of target HW architectures

WHERE

Intel Thread Building Blocks

Intel Thread Building Blocks

A C++ template library
for parallel programming

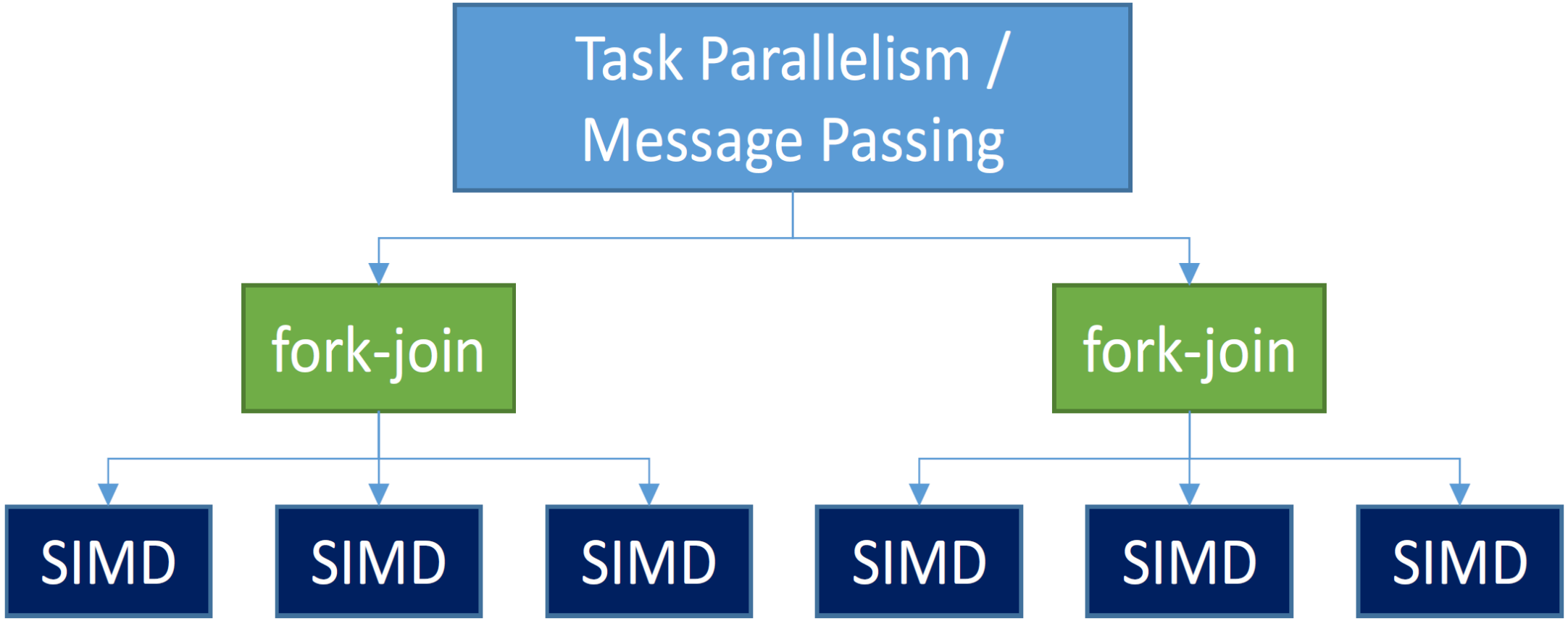


Based on

- Parallel algorithms and data structures
- Task scheduling and work stealing
- Threads and synchronization primitives

TBBs perspective on SHM apps

Three levels of parallelism in a (shared memory) application



Intel Thread Building Blocks

Implements a relaxed sequential execution model

- use as much parallelism as possible, but still be able to run in a single thread
- the sequential execution does not need to be super efficient
- vs. programs that require parallelism to run correctly (e.g., producer-consumer)

Intel Thread Building Blocks

Generic Parallel Algorithms

Efficient scalable way to exploit the power of multi-core without having to start from scratch.

Flow Graph

A set of classes to express parallelism as a graph of compute dependencies and/or data flow

Concurrent Containers

Concurrent access, and a scalable alternative to serial containers with external locking

Synchronization Primitives

Atomic operations, a variety of mutexes with different properties, condition variables

Task Scheduler

Sophisticated work scheduling engine that empowers parallel algorithms and flow graph

Thread Local Storage

Unlimited number of thread-local variables

Threads

OS API wrappers

Miscellaneous

Thread-safe timers and exception classes

Memory Allocation

Scalable memory manager and false-sharing free allocators

TBB Flow Graph

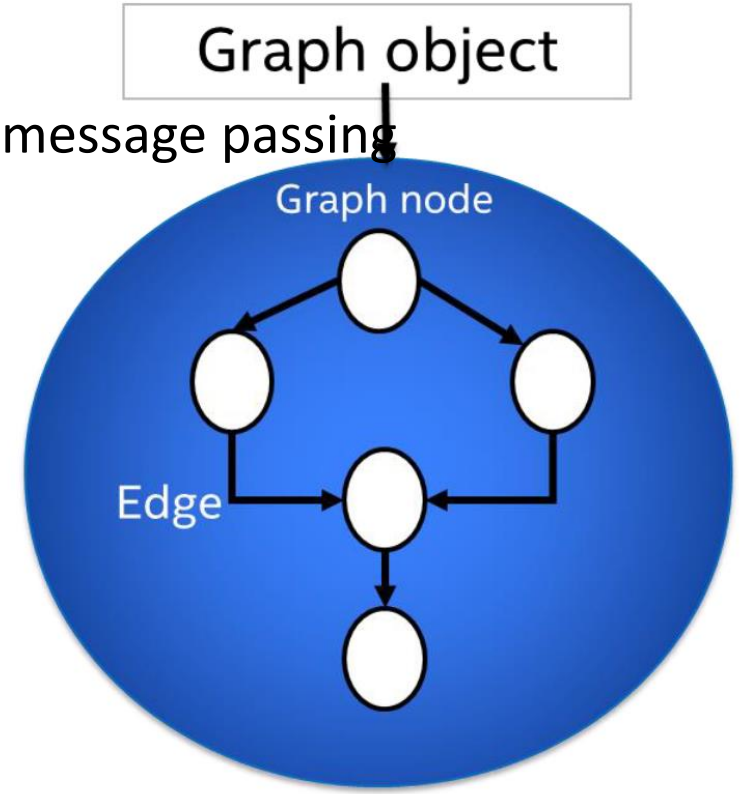
Developers exploit parallelism at the design level

A high-level design concept for shared memory applications

- Parallelism as large computations that communicate by explicit message passing

Allows an efficient implementation of data flows and the representation of dependencies

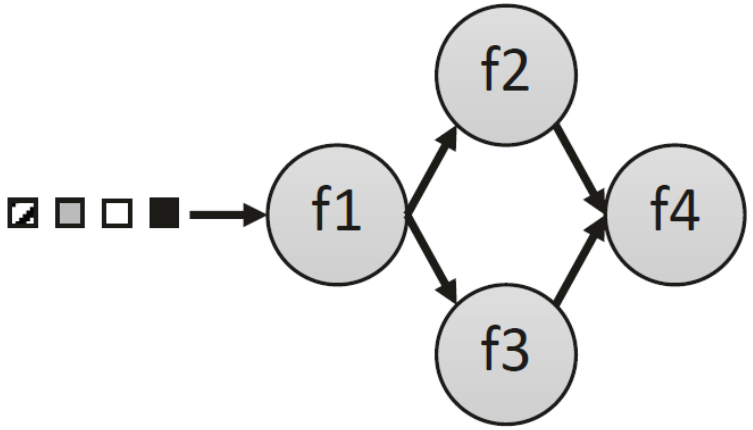
- Graph structure can be used at runtime to schedule the computations in parallel



TBB Flow Graph

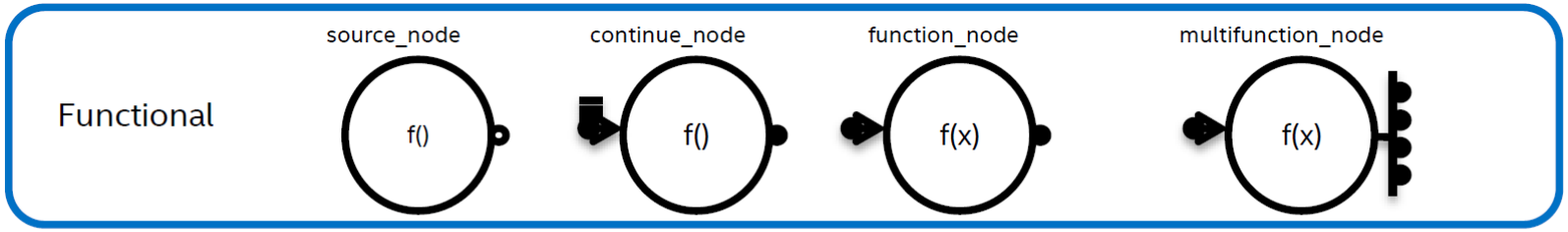
```
while (img = getImage()) {
   x = f1(img);
   y = f2(x);
   z = f3(x);
   f4(y,z);
}
```

TBB Flow Graph



TBB Flow Graph

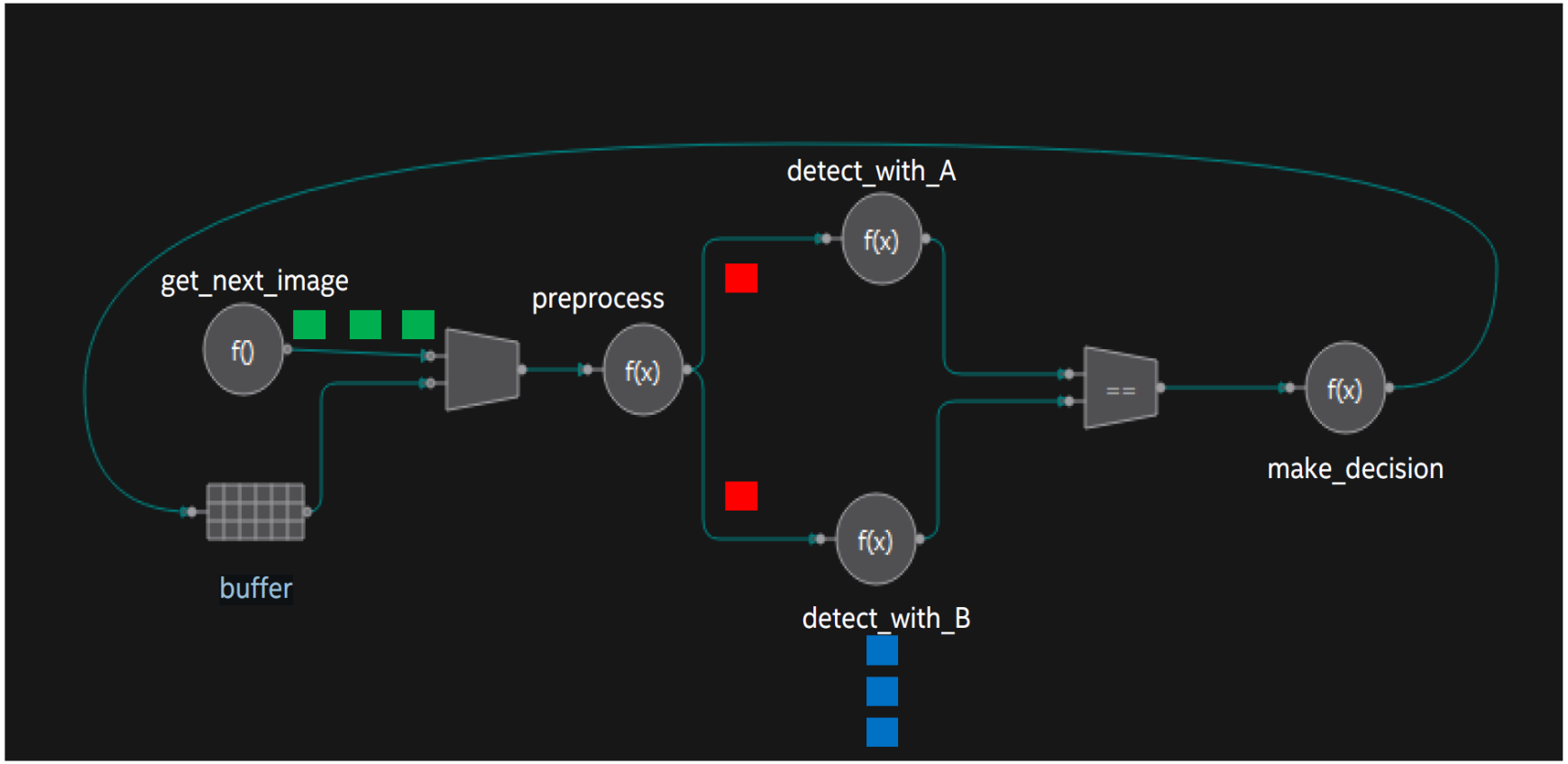
Support of various node types



TBB Flow Graph

Ability to model various parallel concepts

- Pipelining
- Task parallelism
- Data parallelism



Intel Thread Building Blocks

Generic Parallel Algorithms

Efficient scalable way to exploit the power of multi-core without having to start from scratch.

Flow Graph

A set of classes to express parallelism as a graph of compute dependencies and/or data flow

Concurrent Containers

Concurrent access, and a scalable alternative to serial containers with external locking

Synchronization Primitives

Atomic operations, a variety of mutexes with different properties, condition variables

Task Scheduler

Sophisticated work scheduling engine that empowers parallel algorithms and flow graph

Thread Local Storage

Unlimited number of thread-local variables

Threads

OS API wrappers

Miscellaneous

Thread-safe timers and exception classes

Memory Allocation

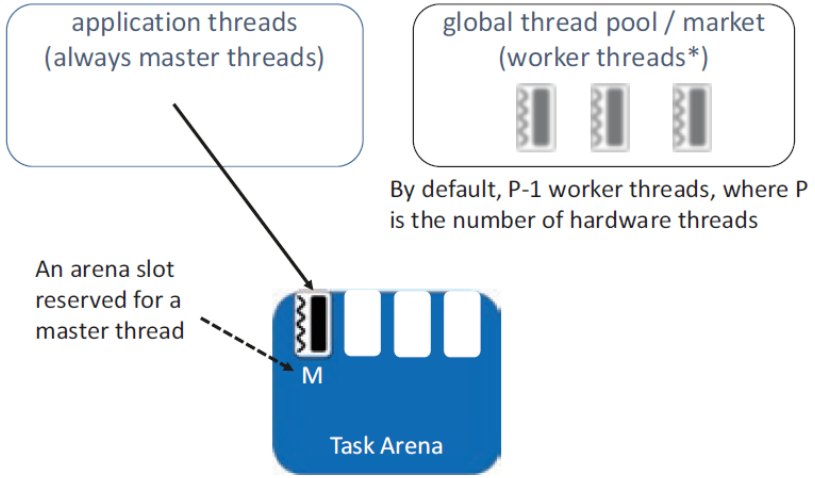
Scalable memory manager and false-sharing free allocators

TBB task scheduling

Based on work-stealing

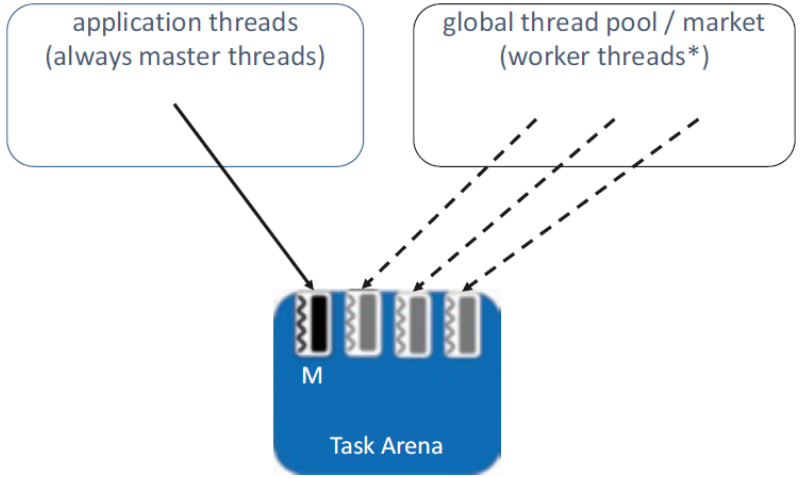
- worker threads actively look for new work when they become idle (cmp. with work-sharing)
- good for dynamic environments with applications that dynamically spawn many tasks

Uses global thread pool (the Market) and Task arenas



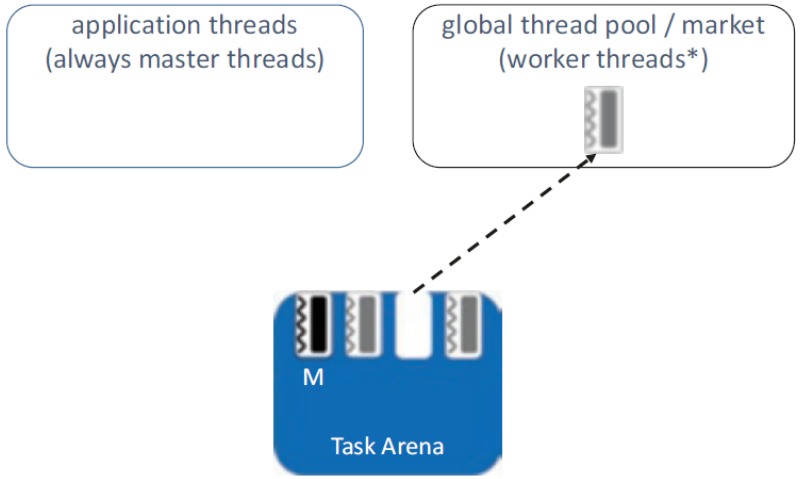
A master thread spawns a task and waits in the arena

(a)



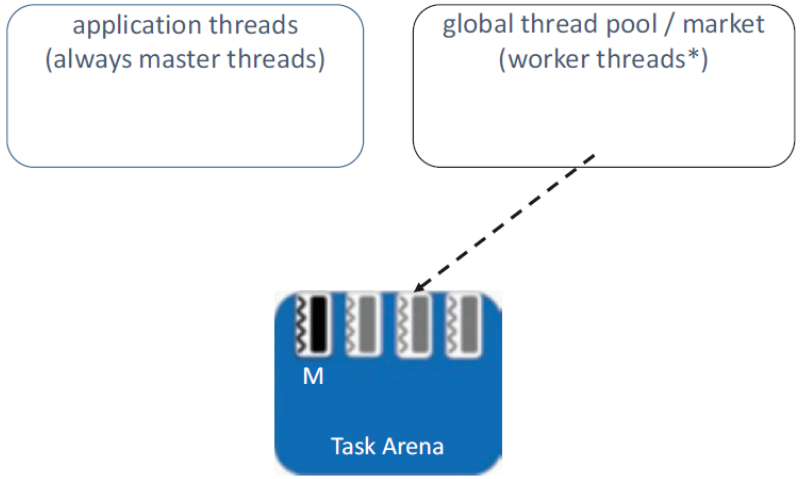
There are available tasks and slots, so workers join the arena

(b)



If a worker becomes idle and can find no more work in the arena, it returns to the global thread pool

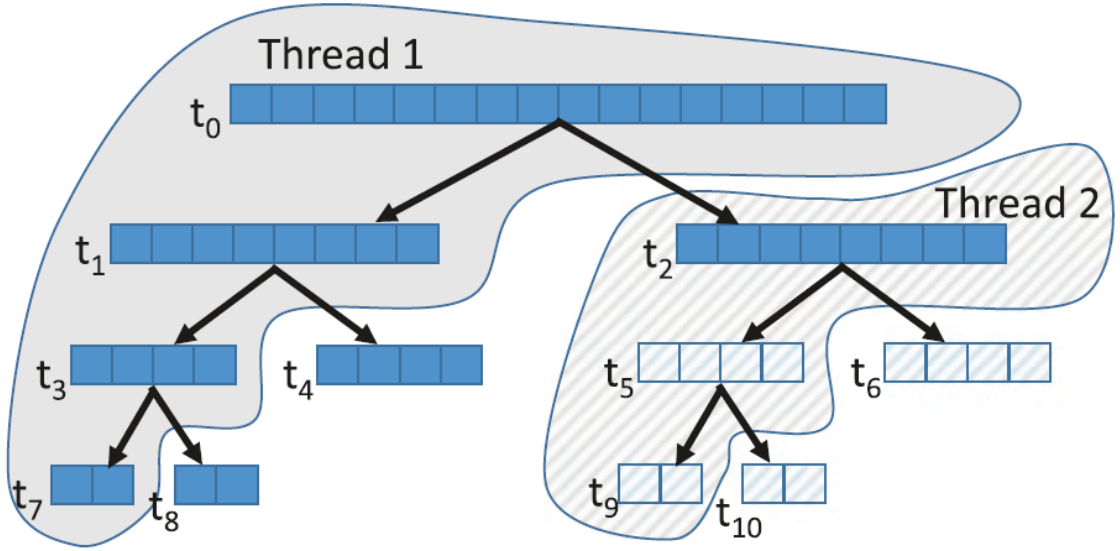
(c)



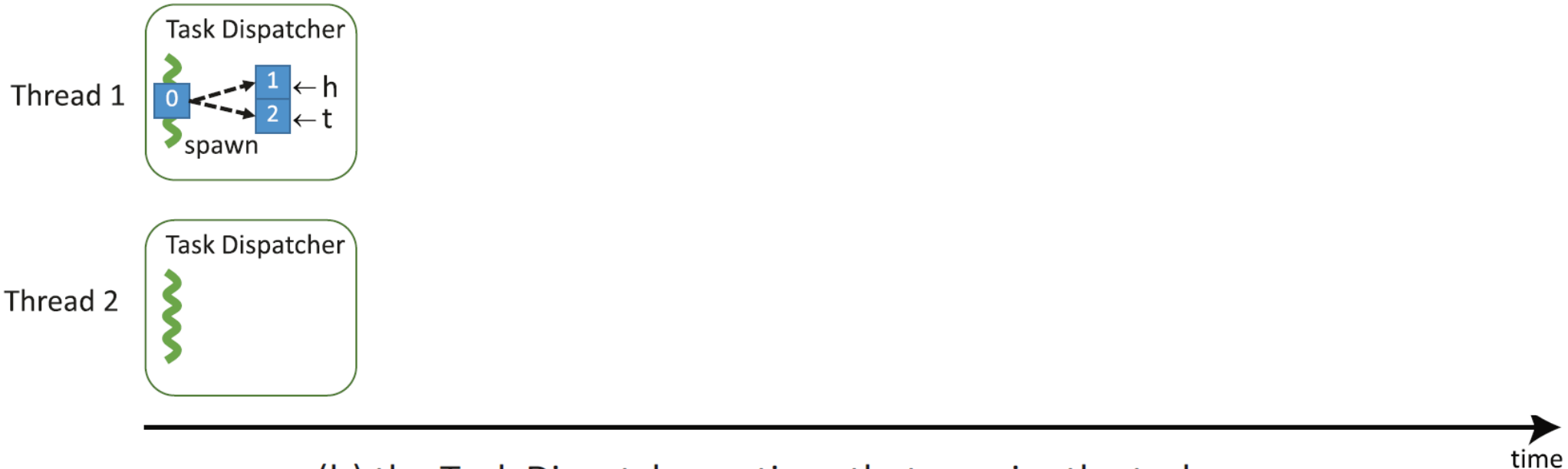
If new tasks become available and a slot is available, the worker will rejoin the task arena.

(d)

TBB task scheduling



(a) tasks as distributed by work stealing across two threads



(b) the Task Dispatcher actions that acquire the tasks

Parallel Matlab

Parallelism in Matlab

Matlab is a high-level language and interactive environment for numerical computation.

Support for parallel programming via the Parallel Computing Toolbox

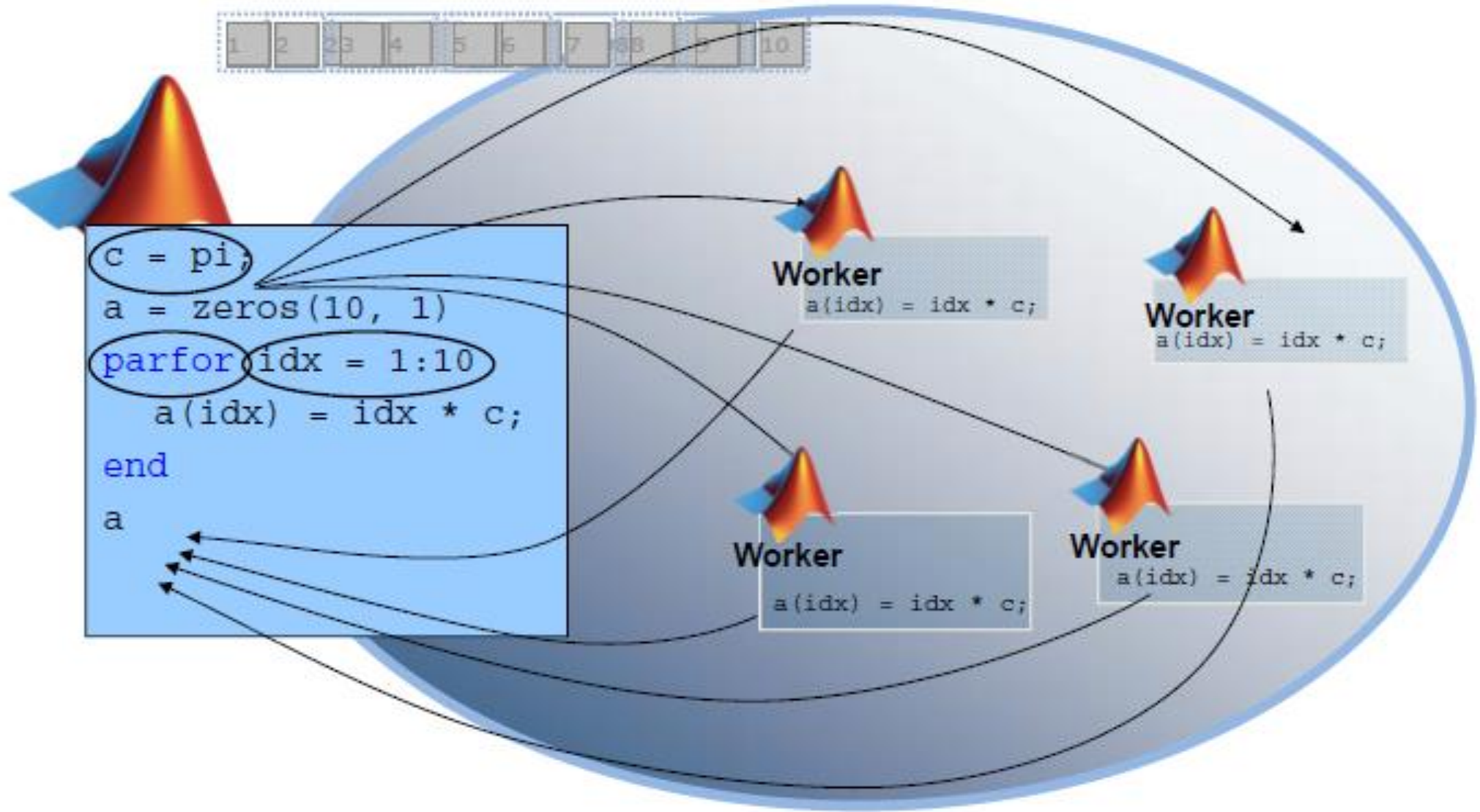
- runs on multicore processors, GPUs, and computer clusters
- supports parallel for-loops, special array types, and parallelized numerical algorithms

Parallelism in Matlab

Basic elements of a parallel computation within Matlab

- *client* - a Matlab session that submits a job
- *communicate job* - a job composed of communicating tasks
- *independent job* - a job composed on independent tasks
- *lab* - an analogy of worker in other languages
- *pool* - a collection of labs
- *parfor* - parallel for loop
- boradcast, reduction, and sliced variables

Parallelism in Matlab

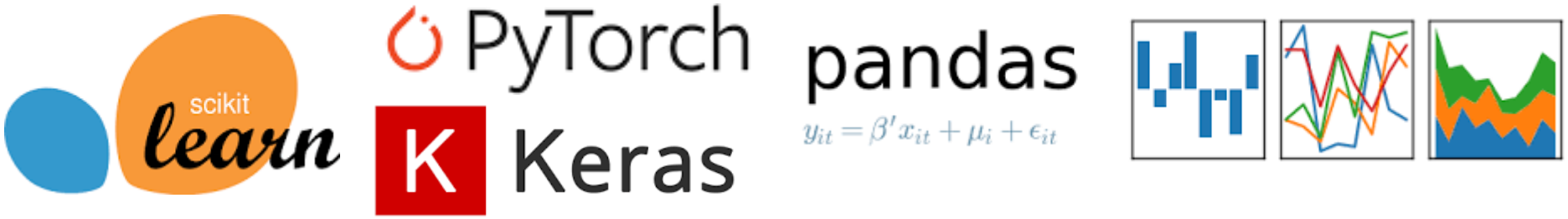


Parallel Python

Parallel Python

Python is an object-oriented, interpreted, and interactive programming language. Easy and fun (2014)

Python is of paramount importance for data science (now)



Parallel Python

Parallel programming in Python

- Not as easy as it could be

Python is an interpreted (JIT compiled) language with automated memory management

- GIL (in CPython) is a mutex that prevents multiple native threads from executing Python bytecodes at once
- it is necessary for CPython's (no thread safe) memory management
- overcome by is multiprocessing