Parallelism for the masses with modern C++

F. Giacomini

INFN-CNAF

Rimini, Workshop CCR, June 2018

Proposal for Parallel Algorithms

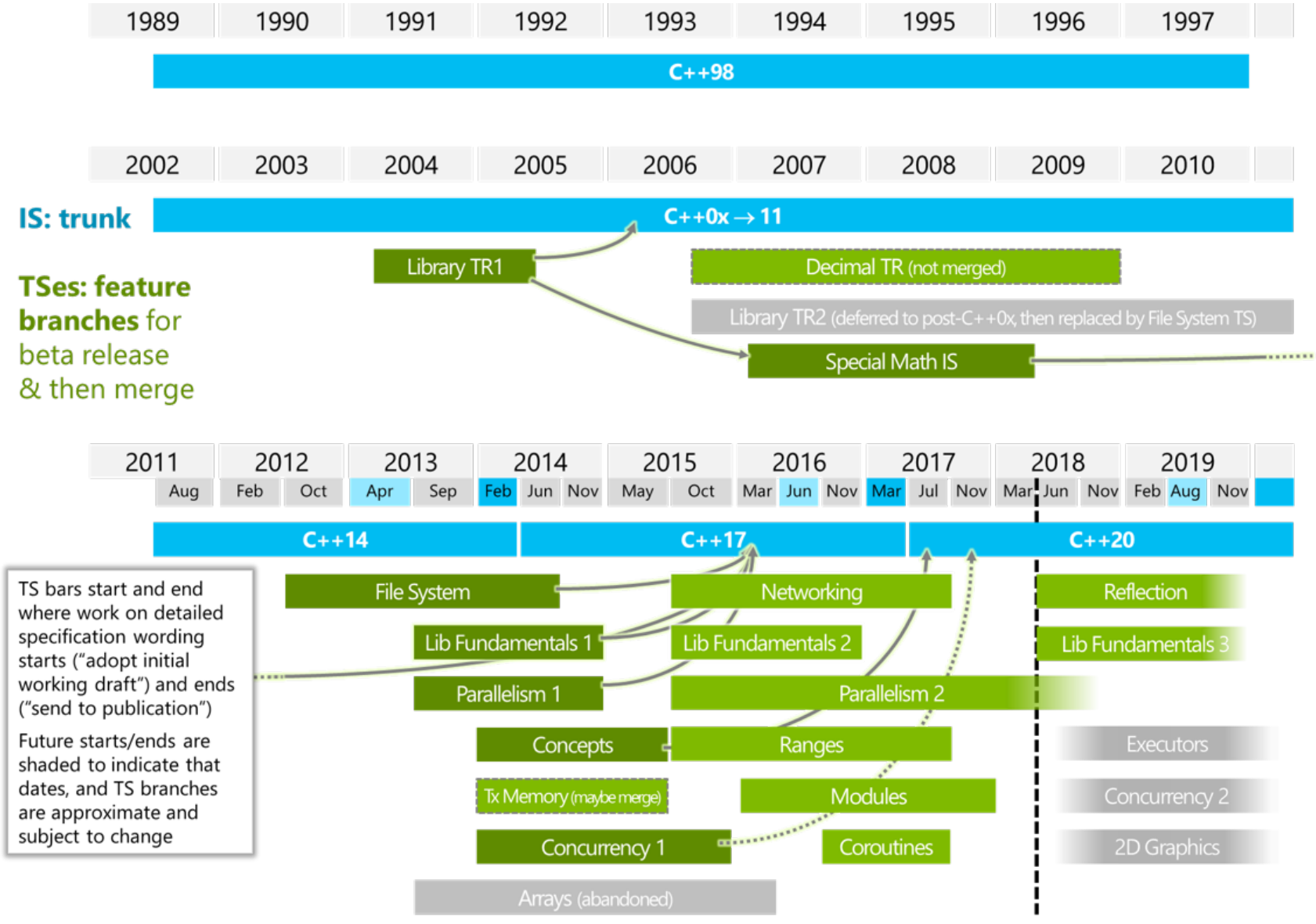
A Parallel Algorithms Library | N3554

Jared Hoberock Jaydeep Marathe Michael Garland Olivier Giroux
Vinod Grover {jhoberock, jmarathe, mgarland, ogiroux, vgrover}@nvidia.com
Artur Laksberg Herb Sutter {arturl, hsutter}@microsoft.com Arch Robison
arch.robison@intel.com

2013-03-15

"This proposal is motivated by a strong desire to provide a standard model of parallelism enabling **performance portability** across the broadest possible range of parallel architectures. For this reason, we have exposed parallelism in the most abstract manner possible in order to avoid presuming the existence of a particular parallel machine model."

C++ timeline


1989
1990
1991
1992
1993
1994
1995
1996
1997
C++98
2002
2003
2004
2005
2006
2007
2008
2009
2010
IS: trunk
C++0x → 11
Library TR1
Decimal TR (not merged)
Library TR2 (deferred to post-C++0x, then replaced by File System TS)
Special Math IS
TSes: feature branches for beta release & then merge
2011
Aug
2012
Feb
Oct
2013
Apr
Sep
2014
Feb
Jun
Nov
2015
May
Oct
2016
Mar
Jun
Nov
2017
Mar
Jul
Nov
2018
Mar
Jun
Nov
2019
Feb
Aug
Nov
C++14
C++17
C++20
TS bars start and end where work on detailed specification wording starts ("adopt initial working draft") and ends ("send to publication")
Future starts/ends are shaded to indicate that dates, and TS branches are approximate and subject to change
File System
Networking
Reflection
Lib Fundamentals 1
Lib Fundamentals 2
Lib Fundamentals 3
Parallelism 1
Parallelism 2
Concepts
Ranges
Executors
Tx Memory (maybe merge)
Modules
Concurrency 2
Concurrency 1
Coroutines
2D Graphics
Arrays (abandoned)

Live coding demo

“Talk is cheap, show me the code”

The path to parallelism

Raw `for` loop $\rightarrow$ Standard algorithm $\rightarrow$ Parallel algorithm

Execution policies

An execution policy indicates the kinds of parallelism allowed in the execution of an algorithm and expresses the consequent requirements on the element access functions

`seq` execution may not be parallelized

`par` execution may be parallelized

- access functions have to be thread-safe, i.e. no data race

`par_unseq` execution may be parallelized and vectorized

- access functions have to be thread- and vectorize-safe, e.g. no memory allocation and deallocation, no synchronization
- for GPUs