



The FASThread Methodology for Fast and Reliable Threading

**Per Stenstrom
CTO**

Nema Labs

Lindholmen Science Park, Goteborg, Sweden

Nema Labs Proprietary and Confidential Data

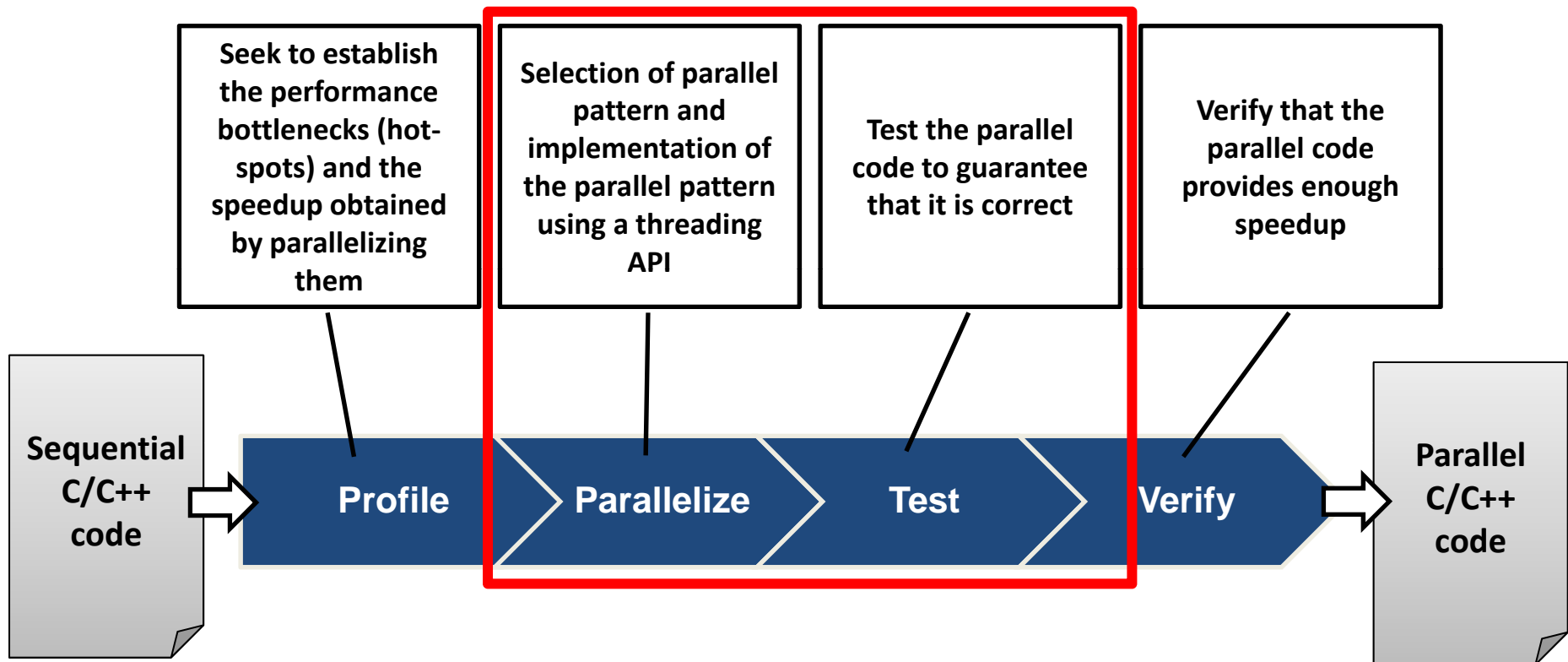
About Nema Labs



- **Mission: Easy-to-use threading tools for C/C++ programs**
- **A spin-out from research at Chalmers University of Technology**
- **Founded in 2007**
- **Backed by venture capital investments from Volvo Technology Transfer and Chalmers Innovation Seed Fund**
- **FASThread, the first product, soon to roll out**
- **Several international patent filings on key technology**

- **What makes threading challenging**
- **FASThread: Fast and reliable threading**
- **FASThread case study**
- **FASThread walk through**
- **Summary**

The Major Obstacles in Parallelization



- **These steps are particularly difficult, labor-intensive, and error-prone**

Parallelization Obstacles 1(2)



Profile

Parallelize

Test

Verify

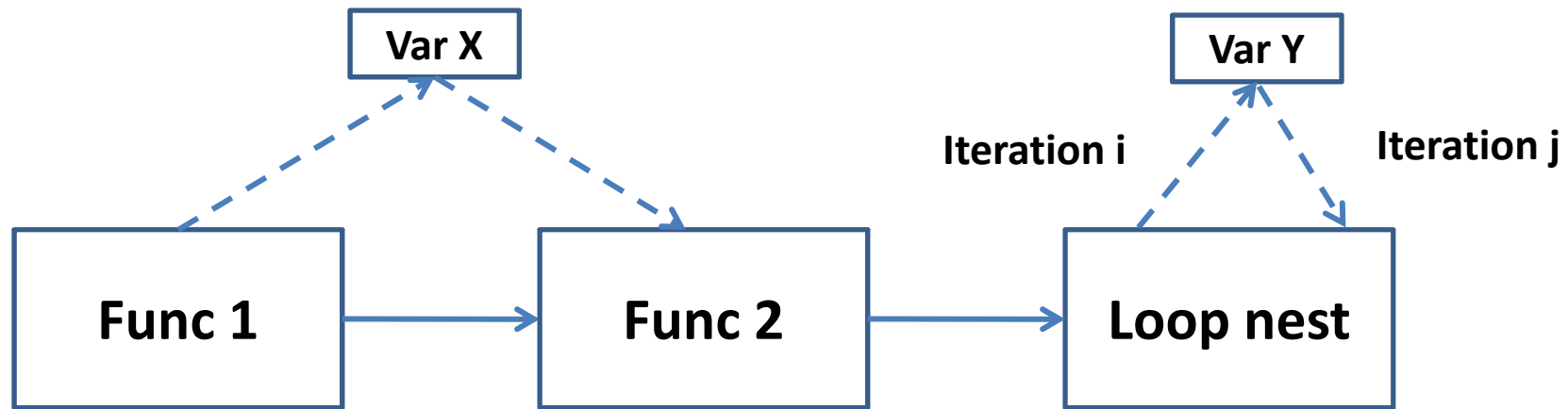
Objective

- Use a threading API to implement the pattern chosen (e.g. function pipeline, task, or data-level parallelism)

Issues

- Which threading API should I choose among the many offered?
- Portability issues
- Requires a significant effort to be productive
- Guarding against ***data races*** and ***synchronization conditions*** is very difficult and error-prone, especially in C/C++ code

Guarding against data races is important



- Function pipeline must have a predictable data flow
- Loop iterations must be independent
- Hard to analyze, particularly in C/C++ due to pointers

Difficult, time-consuming, and error-prone to get right!

Testing Obstacles



Profile

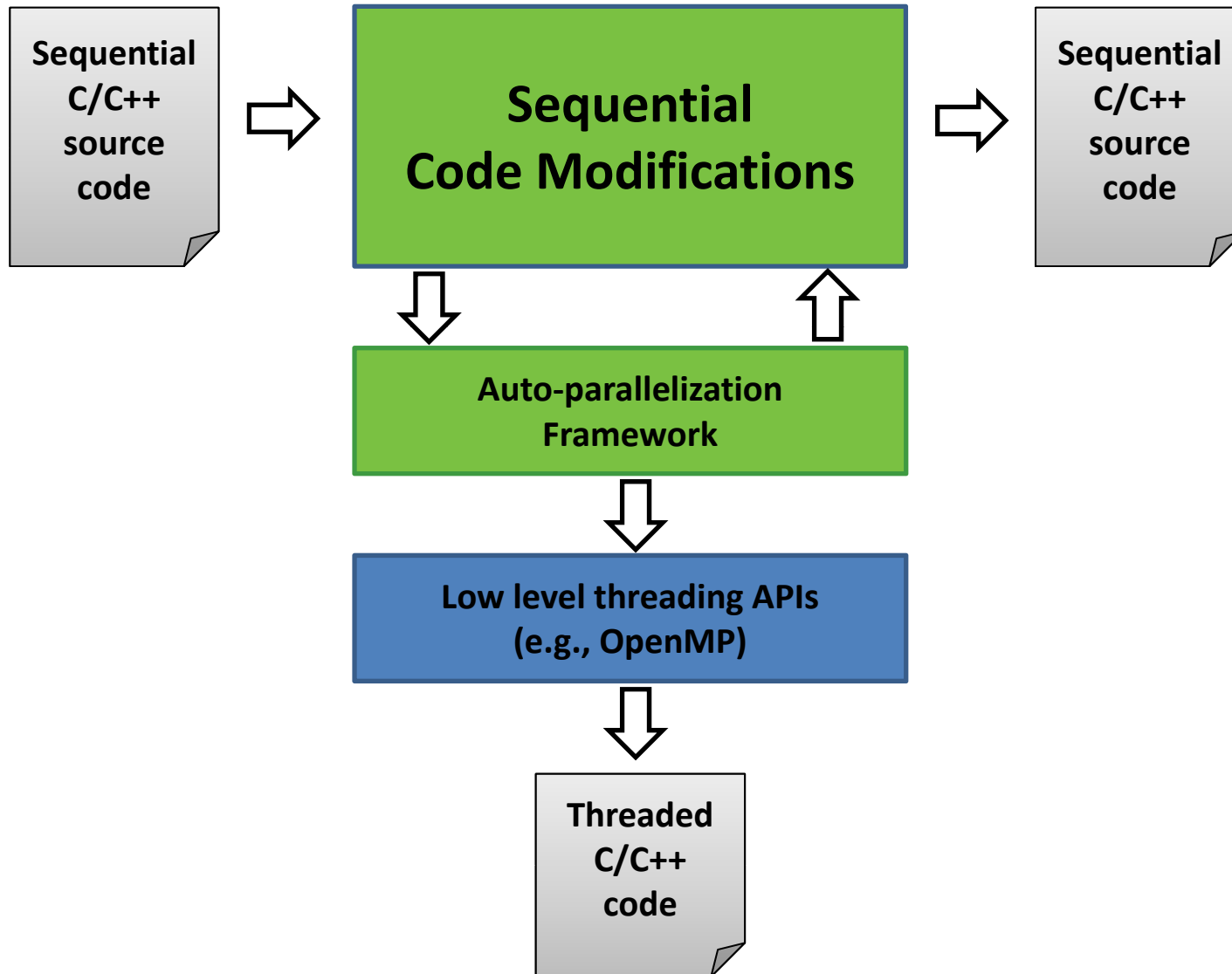
Parallelize

Test

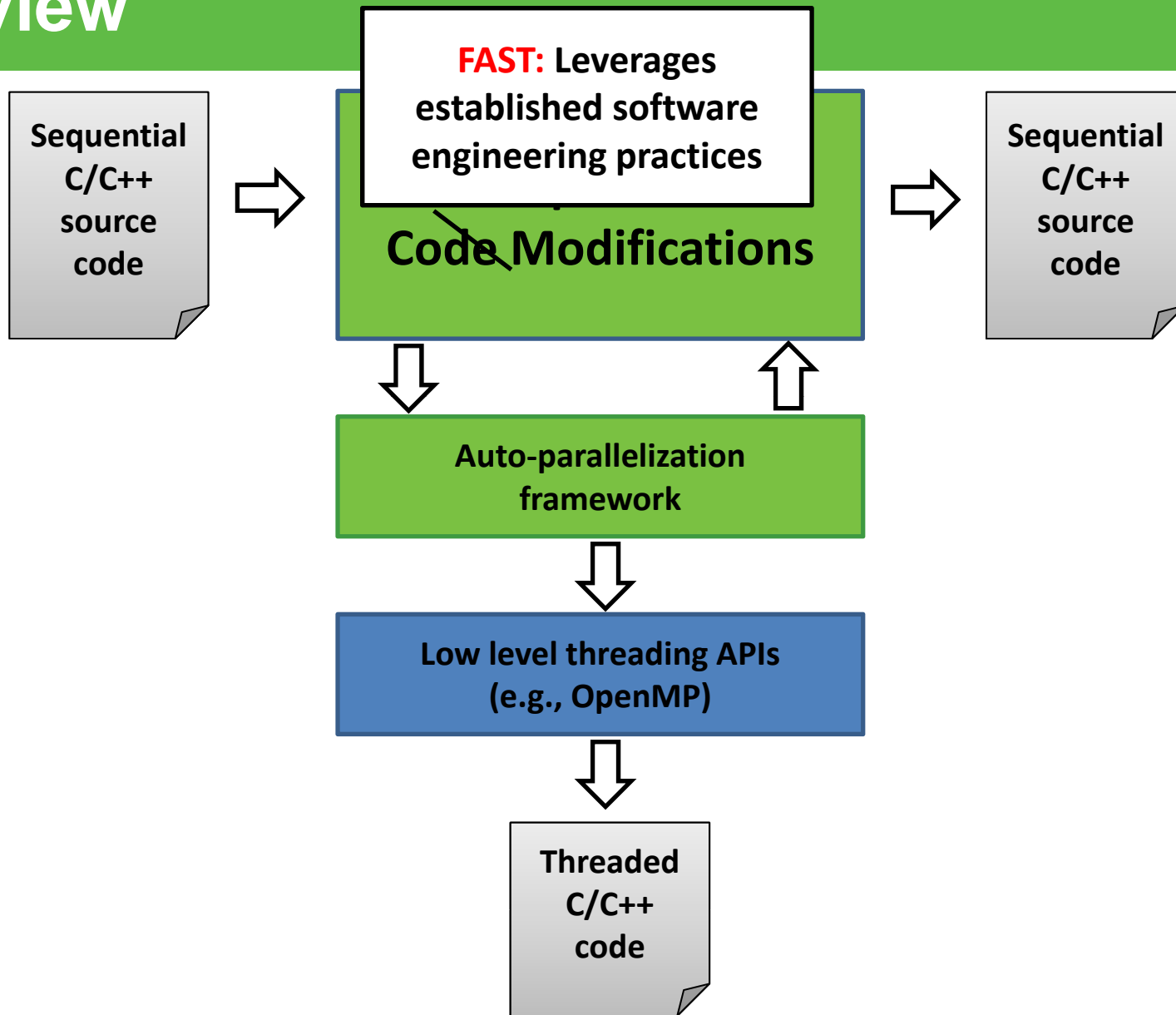
Verify

- **Code segments that run in parallel must be provably independent – no data races**
 - **Trace-based data-race detection technologies are inherently unreliable – input dependent**
 - **Deterministic, replay-based approaches, too slow**
- FASThread guarantees that the parallel code is data-race free – threaded version maintains sequential semantics**

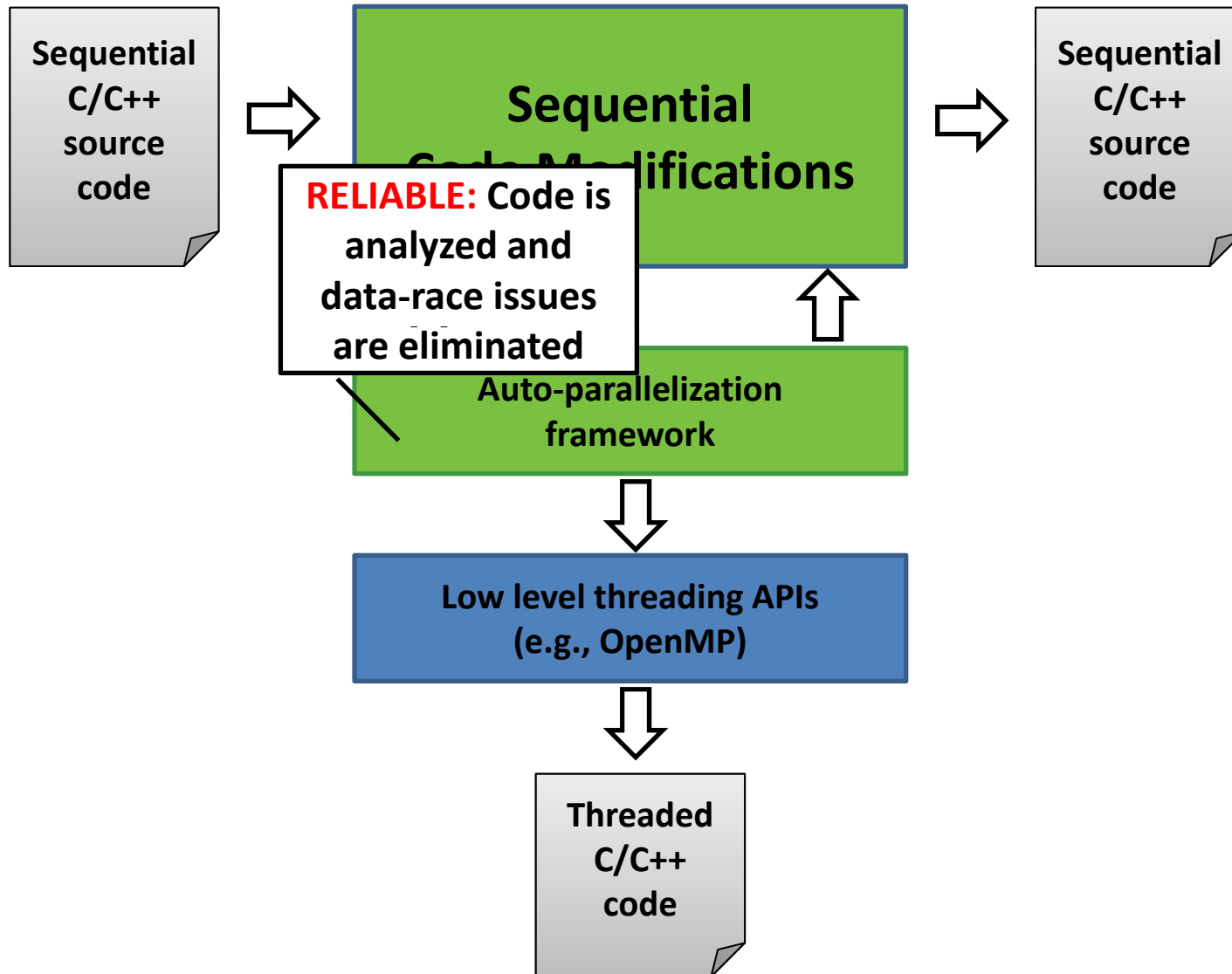
FASThread Threading Methodology Overview



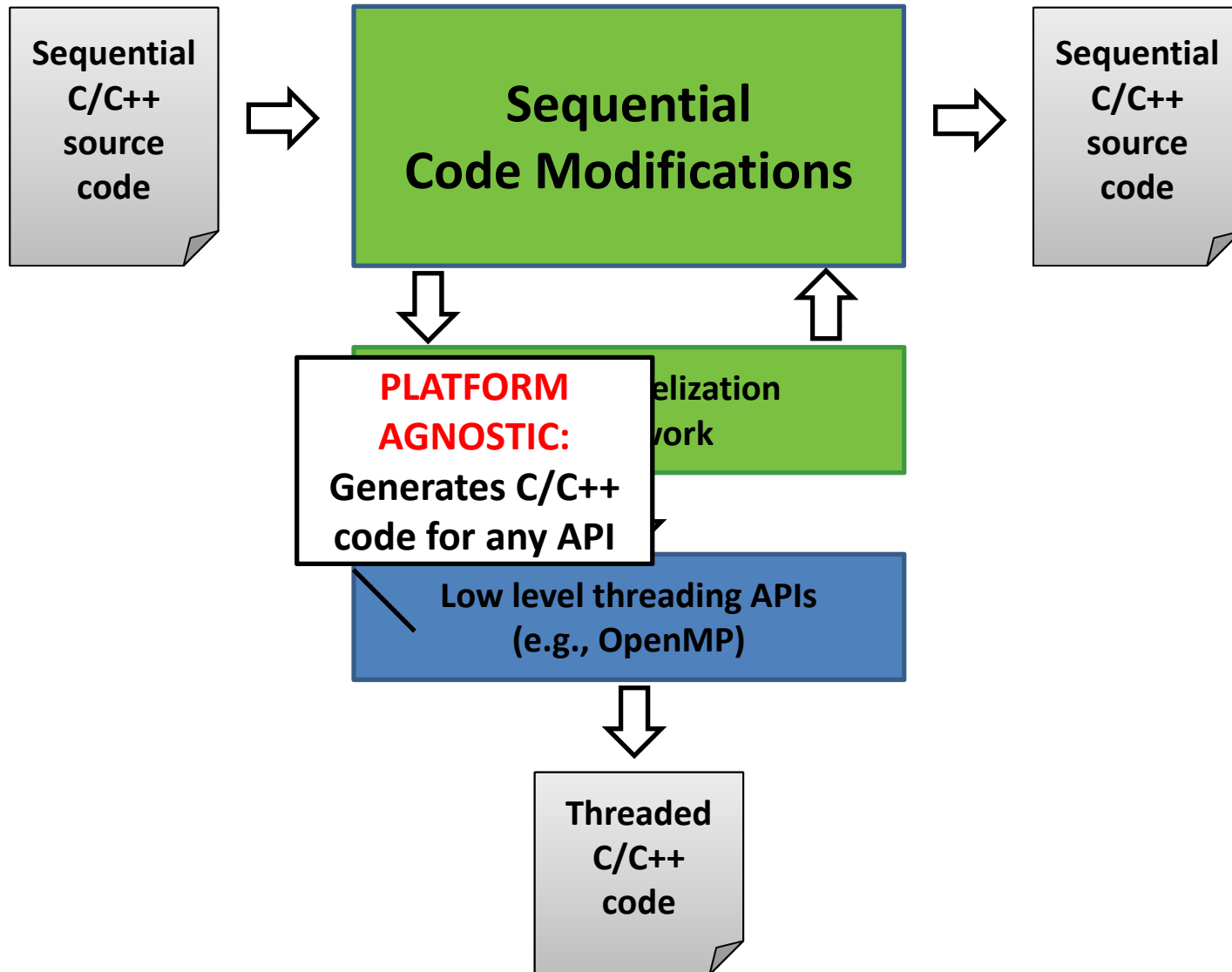
FASThread threading methodology - Overview



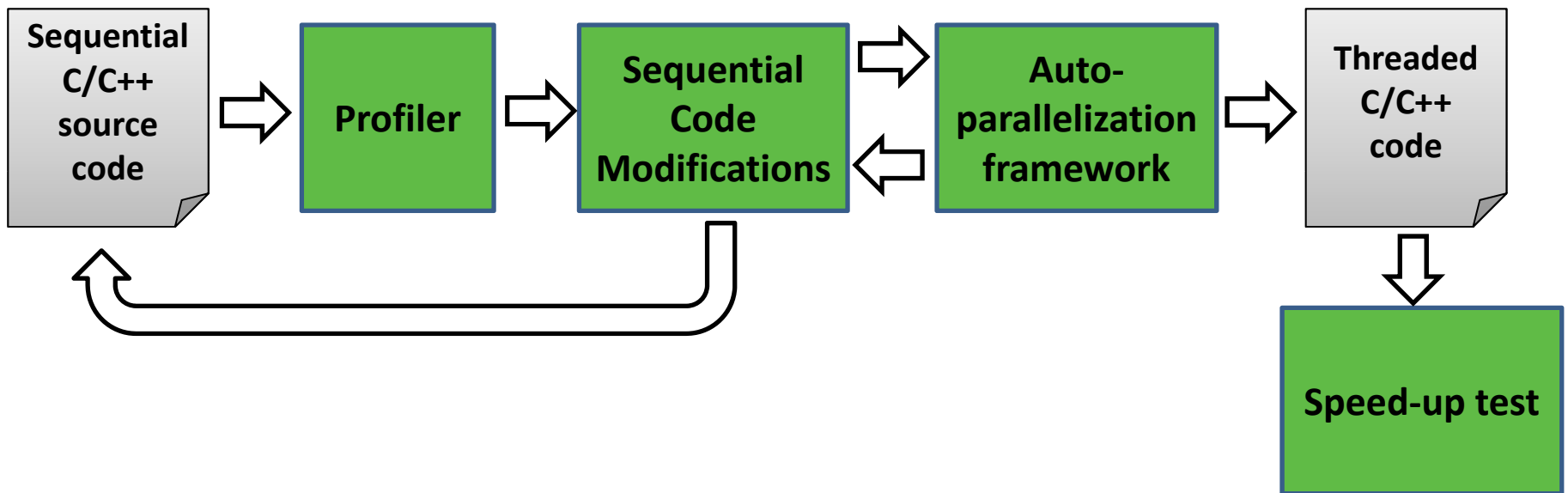
FASThread threading methodology - Overview



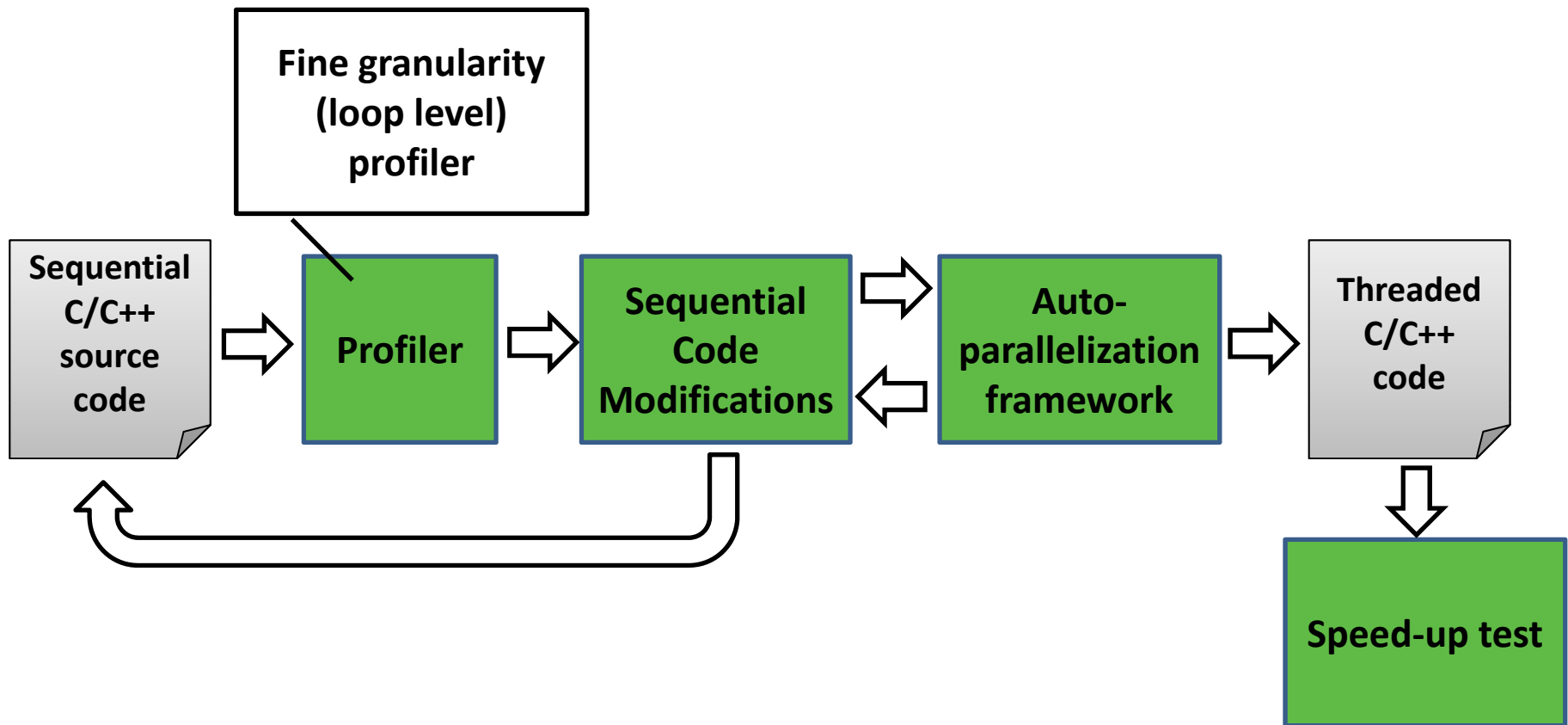
FASThread threading methodology - Overview



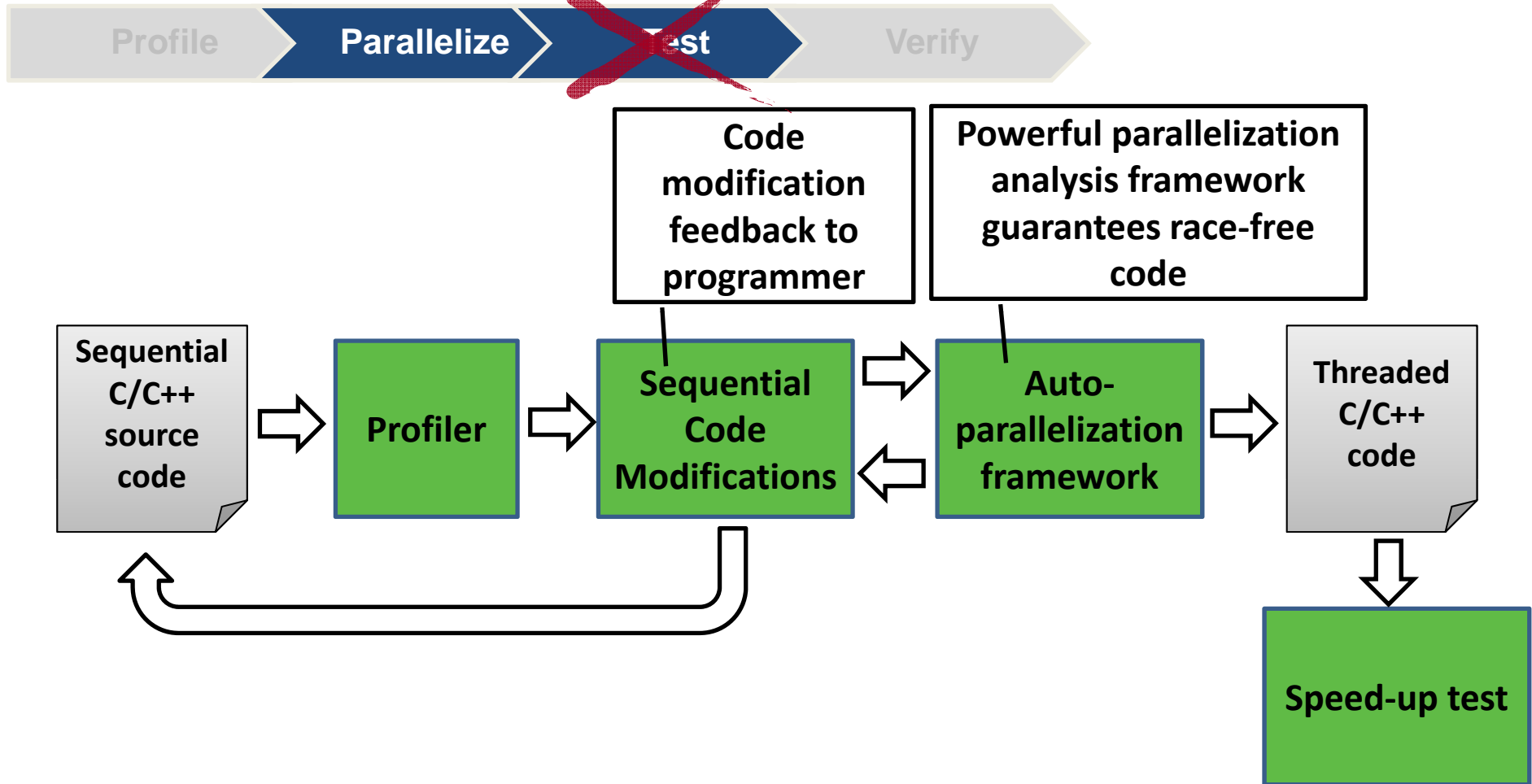
FASThread workflow



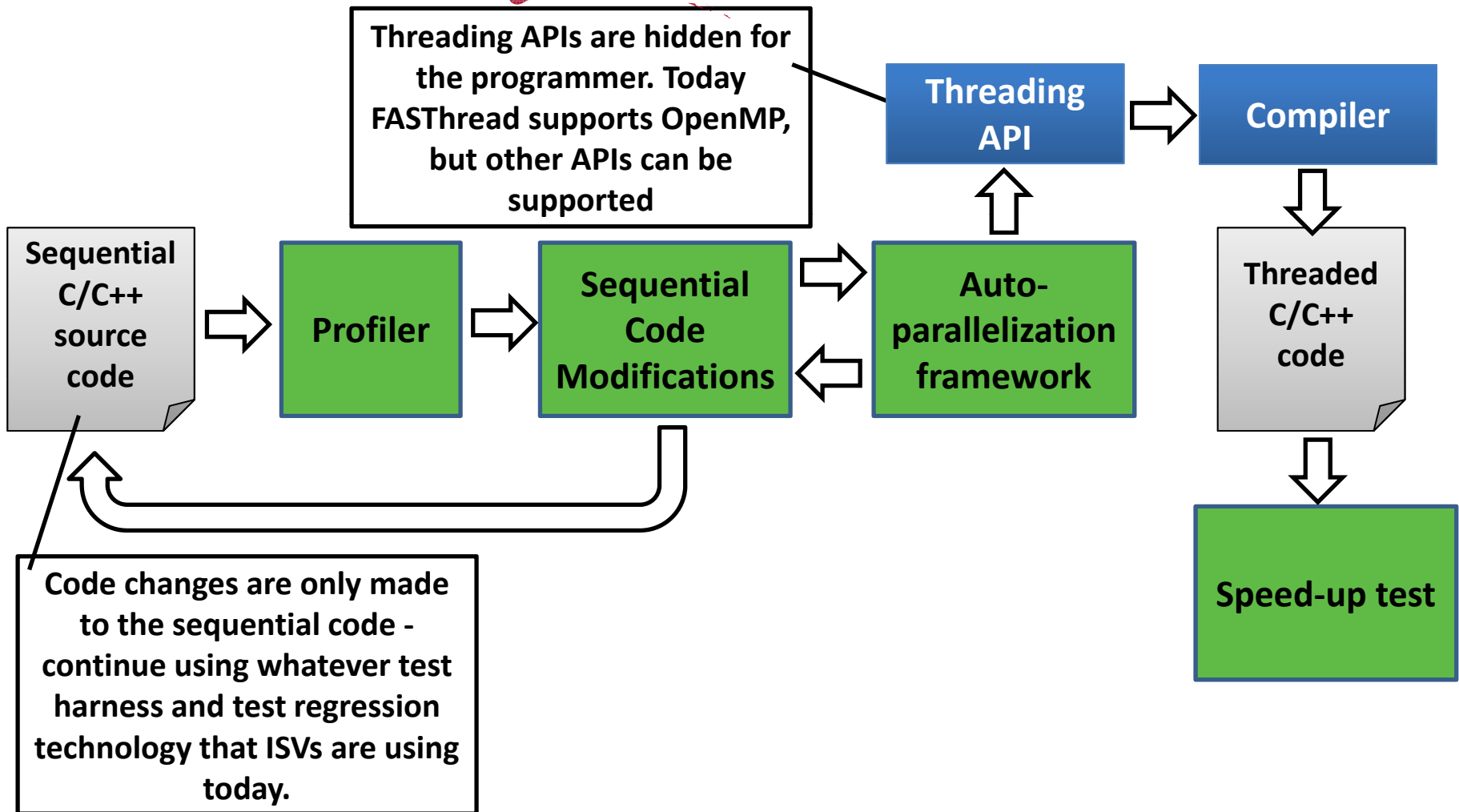
Nema Labs threading methodology – FASThread



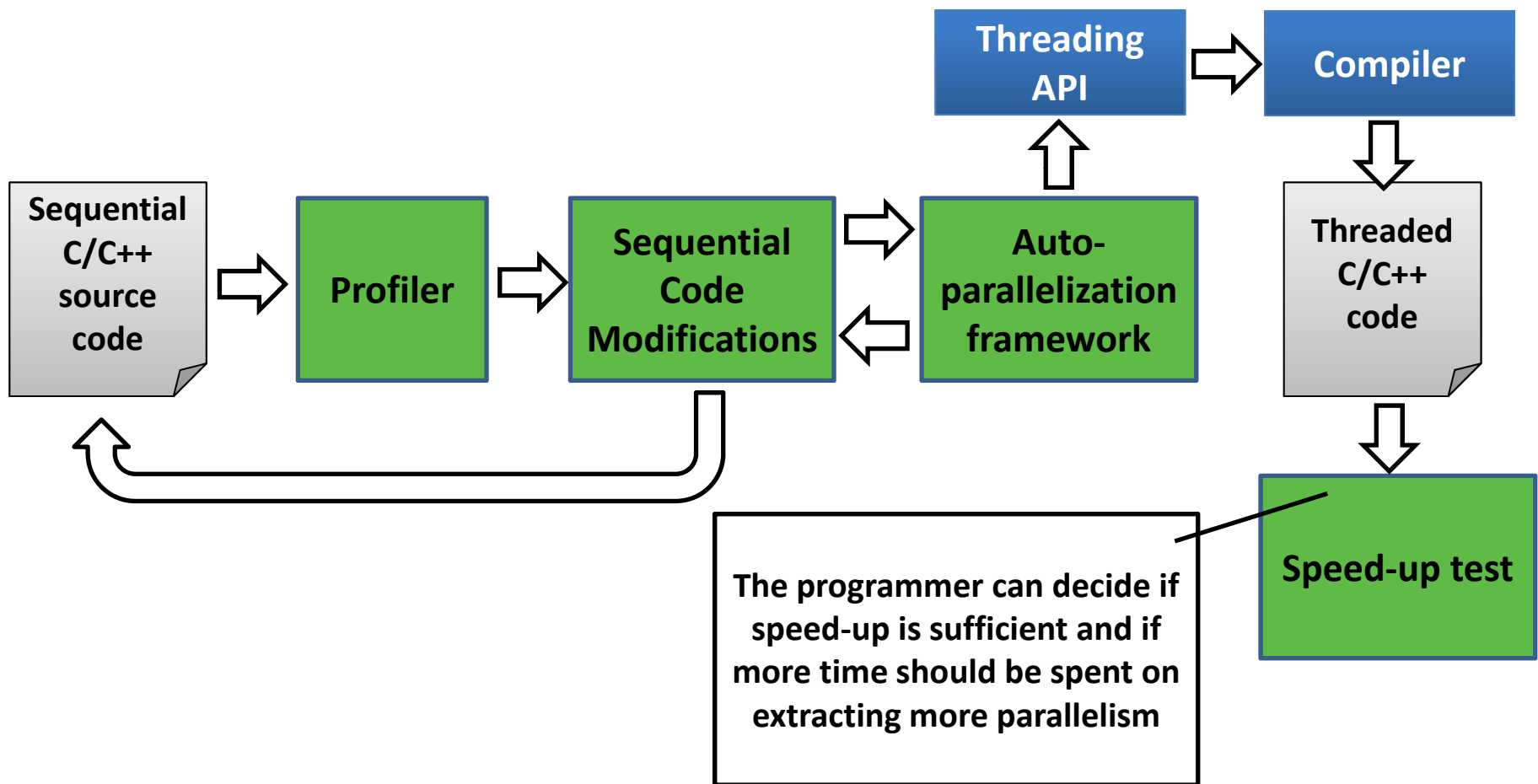
Nema Labs threading methodology – FASThread



Nema Labs threading methodology – FASThread



Nema Labs threading methodology – FASThread



Case studies



Nine applications from the following domains

- **Seismic processing (1 application)**
- **Bioinformatics (1 application)**
- **Biometrical analysis (1 application)**
- **Media/signal processing (6 applications)**

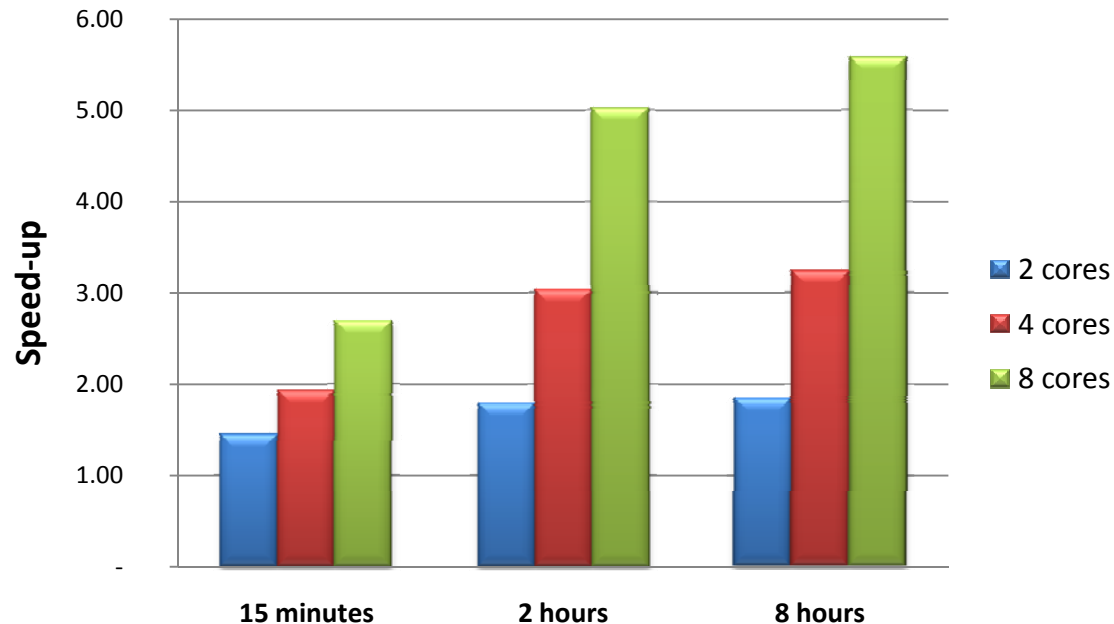
Test system:

- **AMD eight-core system**
- **Compilers: icc with best-case optimization levels**
- **Kubuntu O/S**

Speedup results



Geometric mean speed-up



Significant speedup results in a few hours of "threading"

- FASThread generated results for four out of nine apps immediately < 15 min
- Two out of nine benefitted from code rewriting for another 2 hours
- Another two benefitted from another few hours of code rewriting

NEMA LABS CONFIDENTIAL AND PROPRIETARY

FASThread status

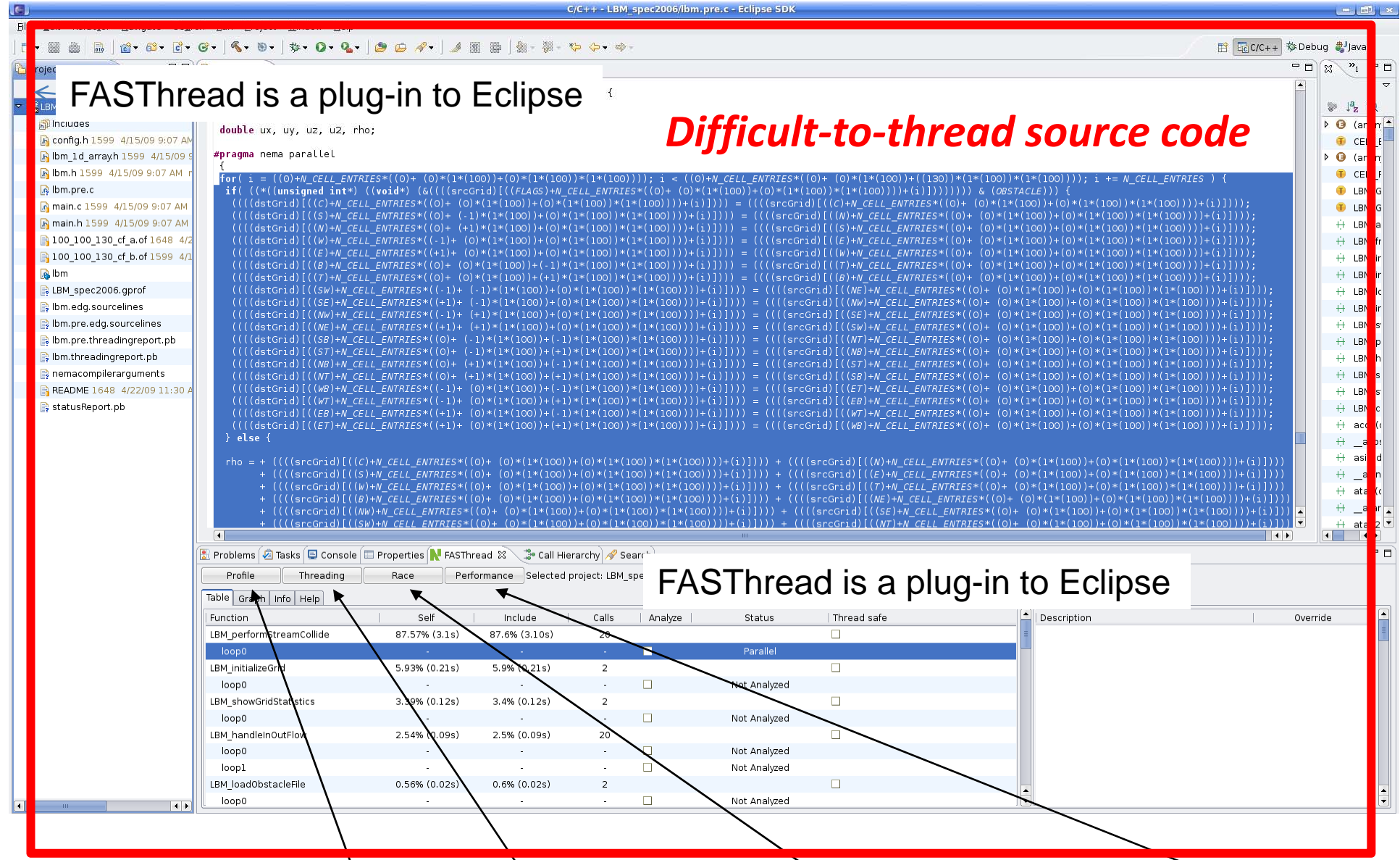


FASThread available for evaluation

- **Supports C**
- **Integrates with Eclipse**
- **O/S support: 32/64 bit Ubuntu/Kubuntu/SLES**
- **Compilers: gcc and icc**
- **Threading API: OpenMP**

Early 2010, FASThread will be released for

- **C++**
- **Windows, Visual studio**



FASTThread is a plug-in to Eclipse

Difficult-to-thread source code

FASTThread is a plug-in to Eclipse

- FASTThread workflow:**
1. Profile
 2. Analyze & thread
 3. Check for race conditions
 4. Performance verification

Summary



- **Threading legacy code in C/C++ is difficult, labor-intensive and error-prone**
- **Nema Labs FASThread threads code in hours that programmers might not even be able to thread, ever**
- **Reliable – no race or synchronization issues thanks to the auto-parallelizing framework**
- **Threading obstacles hidden – easy to learn**
- **Threads fasts – hours, not days or weeks**