St Petersburg University
www.spbu.ru

ON PORTING OF APPLICATIONS TO NEW HETEROGENEOUS SYSTEMS

Alexander Bogdanov
Nikita Storublevtcev
Vladimir Mareev
Denis Manyashin

The Problem

- We have an algorithm.
- We want it to run on GPU.
- ???

Ways to Solve it

- Automatic

  Compiler does everything for us.
- Semi-automatic

  We use libraries and frameworks to do some things for us.
- Manual

  We use CUDA/OpenCL API to do everything ourselves.
- Insane

  Writing in-house GPU drivers.

Automatic Approach

- Easy on the programmer.
- Requires minimal experience and knowledge.
- Allows to easily port existing codebases.
- May produce suboptimal code.
- May not be actually possible in the future.

Semi-Automatic Approach

- Requires learning the library API.
- Third-party dependent.
- May produce sub-optimal code.

Manual Approach

- A lot of effort from programmer.
- Requires learning CUDA/OpenCL API and programming model.
- May be impractical to port existing codebases due to their size.
- Best potential for maximum performance.

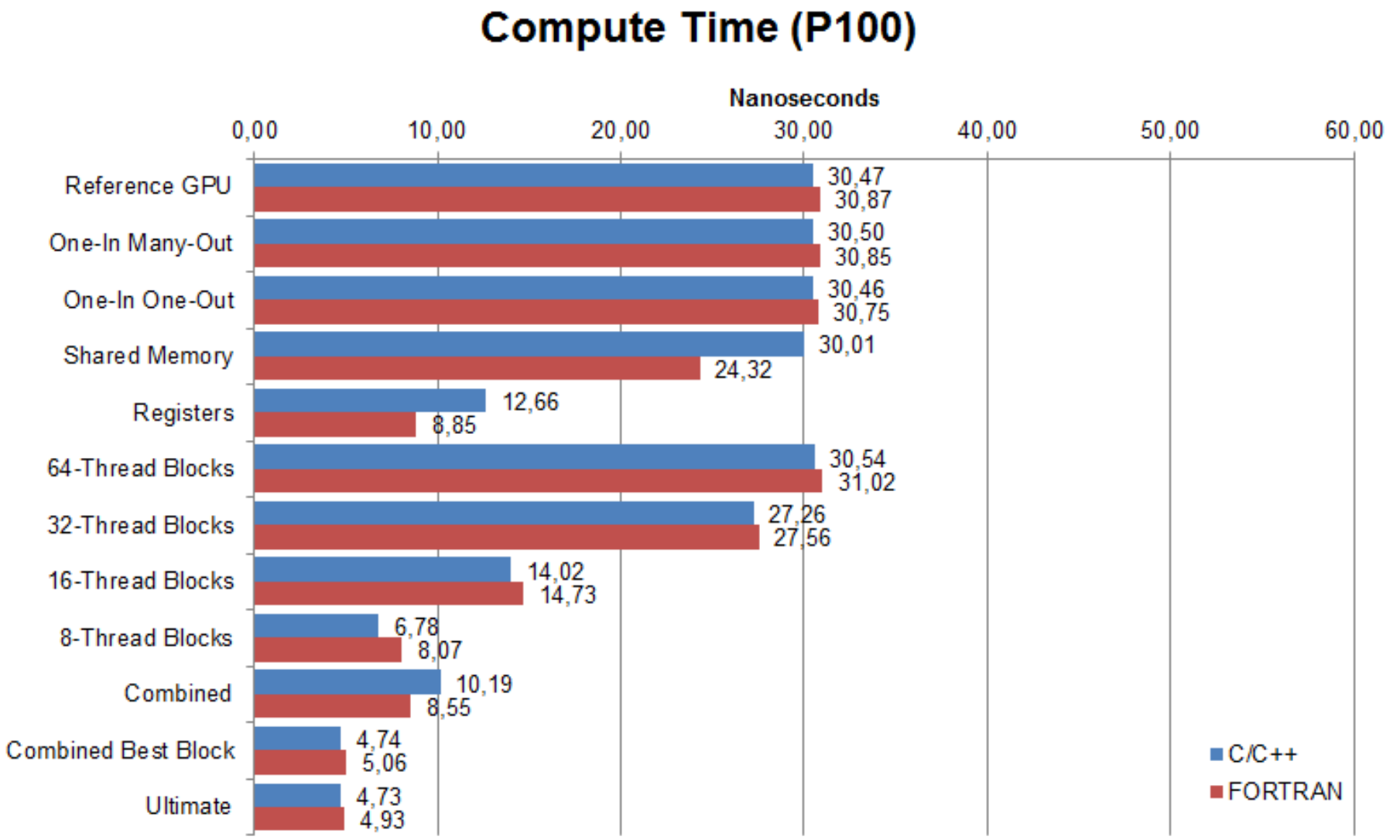
GPGPU Specifics

- RAM-GPU memory transfer is a major bottleneck most of the time.
- GPU uses parallelism to hide the memory access latency.
- Granularity of parallelism needs to be carefully considered for each problem.

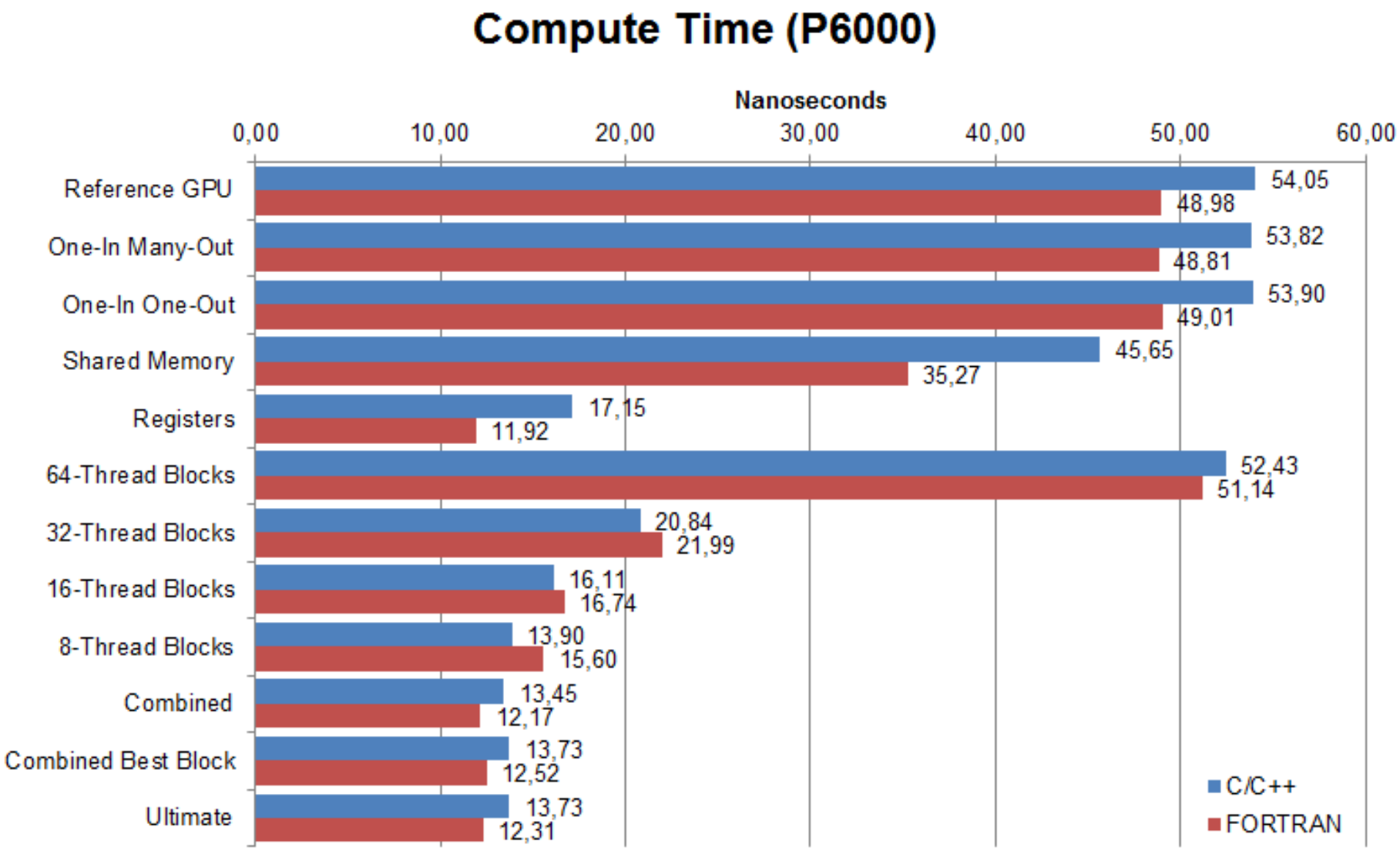
The Hardware

- NVIDIA TESLA P100 GPU, Pascal architecture, 3584 CUDA cores, 1328-1480 MHz clock speed (9340 GFLOPS total), 16 GB global memory
- NVIDIA QUADRO P6000 GPU, Pascal architecture, 3840 CUDA cores, 1417 MHz clock speed (12634 GFLOPS total), 24 GB global memory

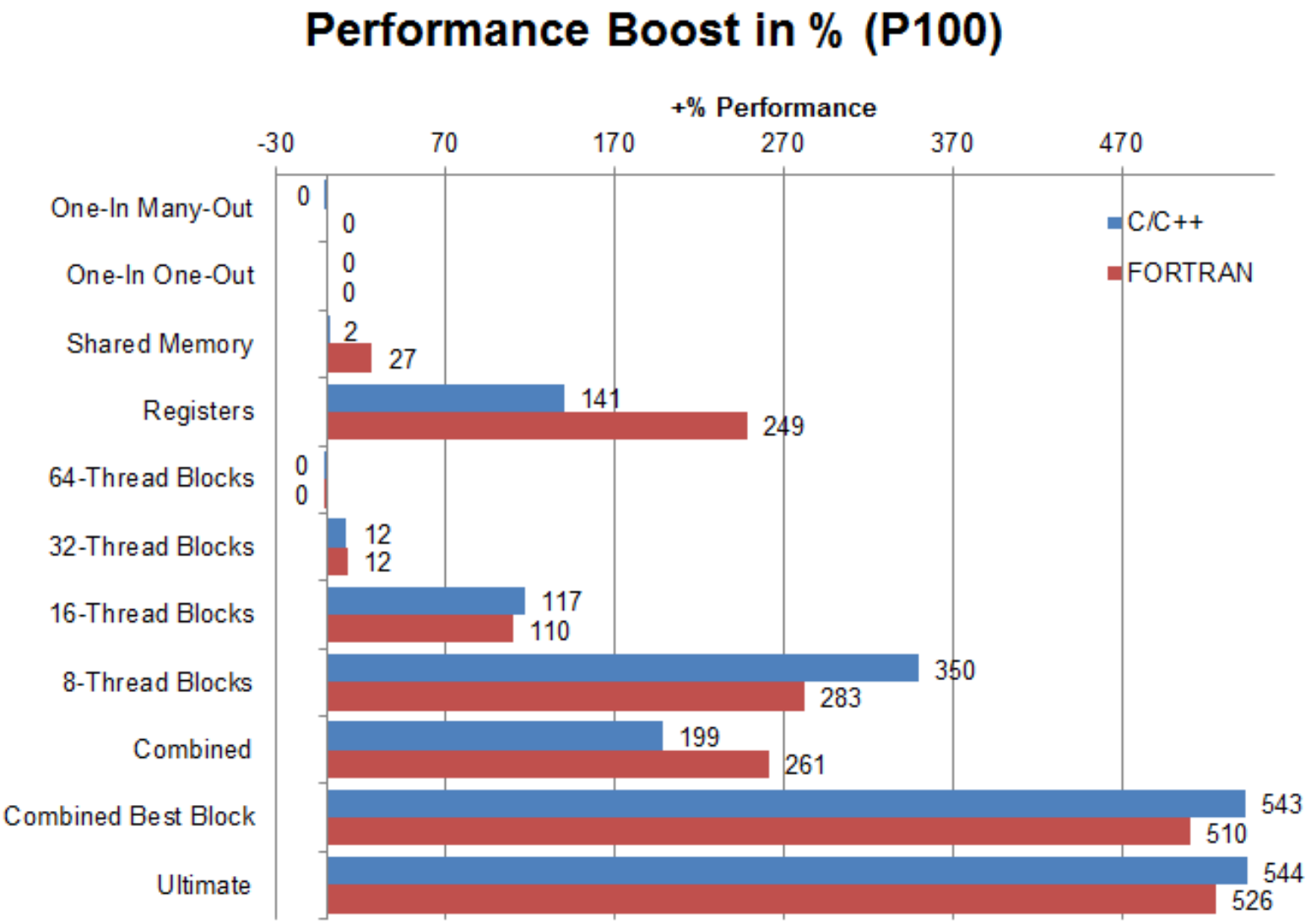
Optimization Methods



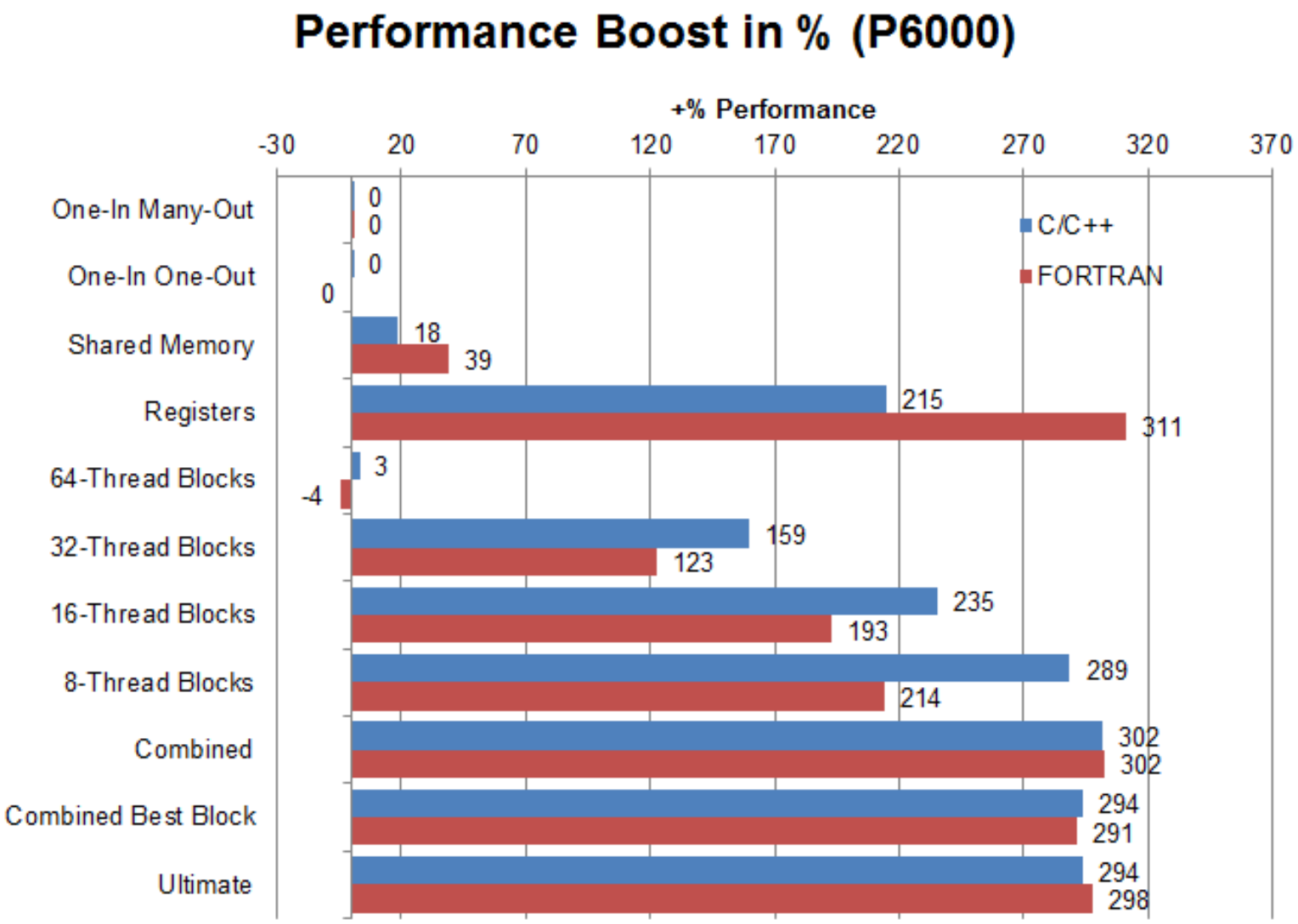
Optimization Methods



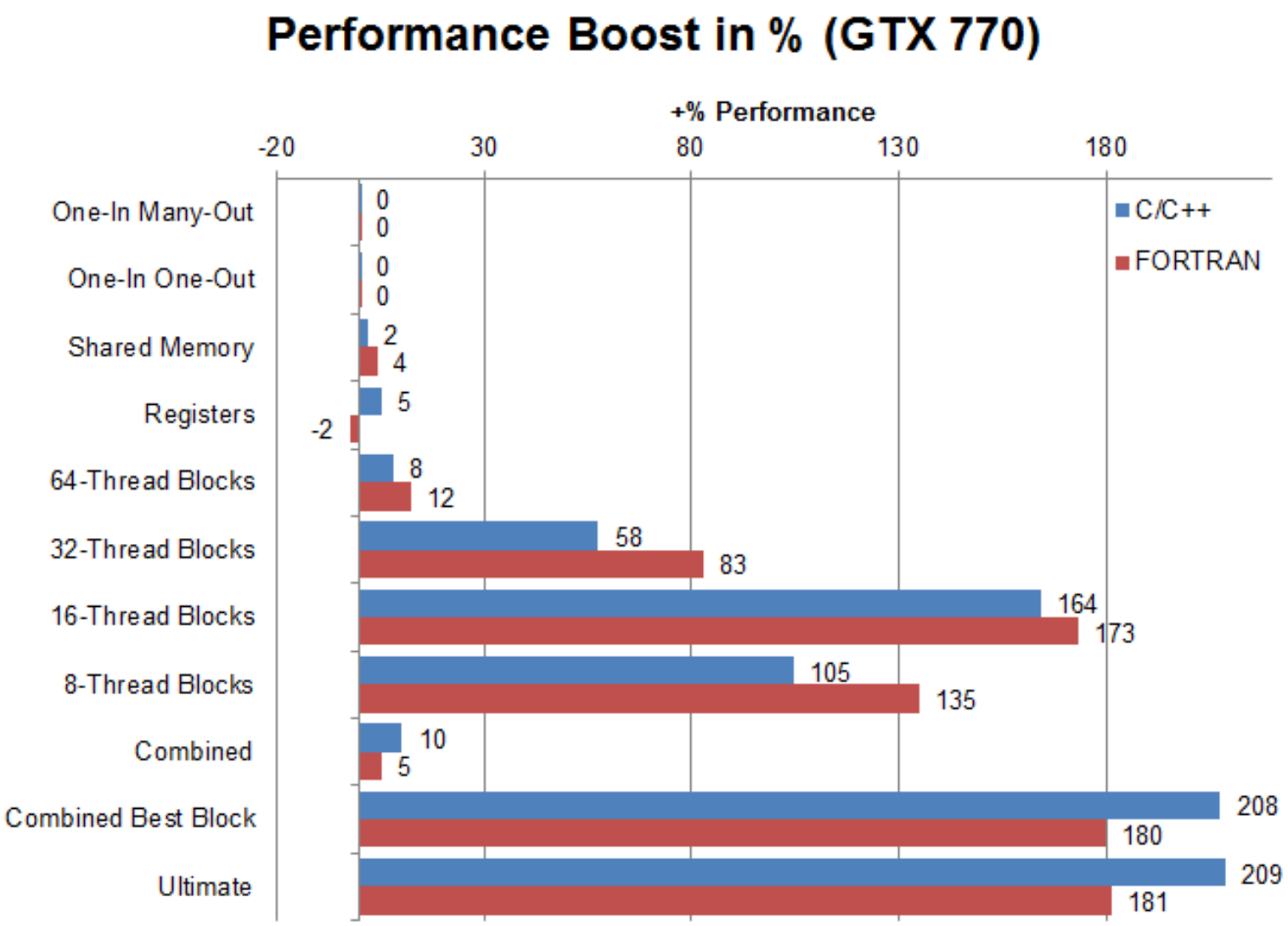
Optimization Methods



Optimization Methods



Optimization Methods



Conclusion

- GPGPU technology has great potential, but it is hard to utilize.
- Application performance is highly dependent on the hardware specifications.
- Effectiveness of optimization methods is also highly dependent on hardware.

Thank you.
Questions?

St Petersburg University

spbu.ru