A RUNTIME-ADAPTABLE INSTRUMENTATION BACK-END FOR SCORE-P

XRay for Score-P

Sebastian Kreutzer, **Paul Adelmann**, Christian Bischof

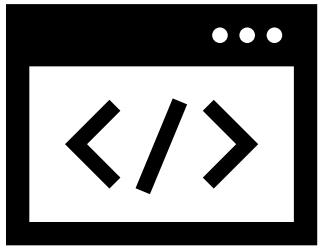
Scientific Computing, TU Darmstadt
{sebastian.kreutzer, christian.bischof}@tu-darmstadt.de
paul.adelmann@stud.tu-darmstadt.de

OUTLINE

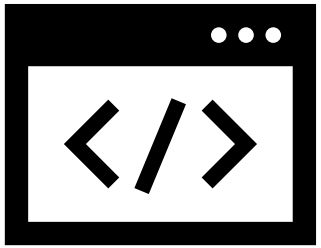
Background — Implementation — Evaluation — Discussion — Outlook — Related work

MOTIVATION & BACKGROUND

WHICH APPLICATION RUNS FASTER?

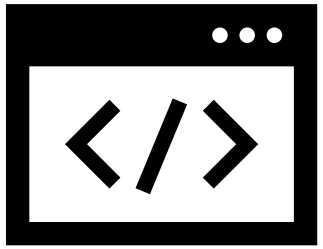


test1

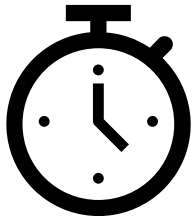


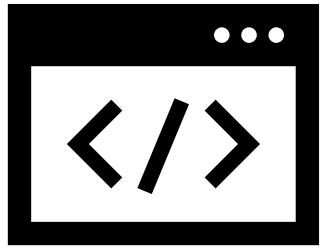
test2

WHICH APPLICATION RUNS FASTER?



test1





test2

Sampling

```
    void work(){
→     // …
→     // …
→     // …
    }
```

Instrumentation

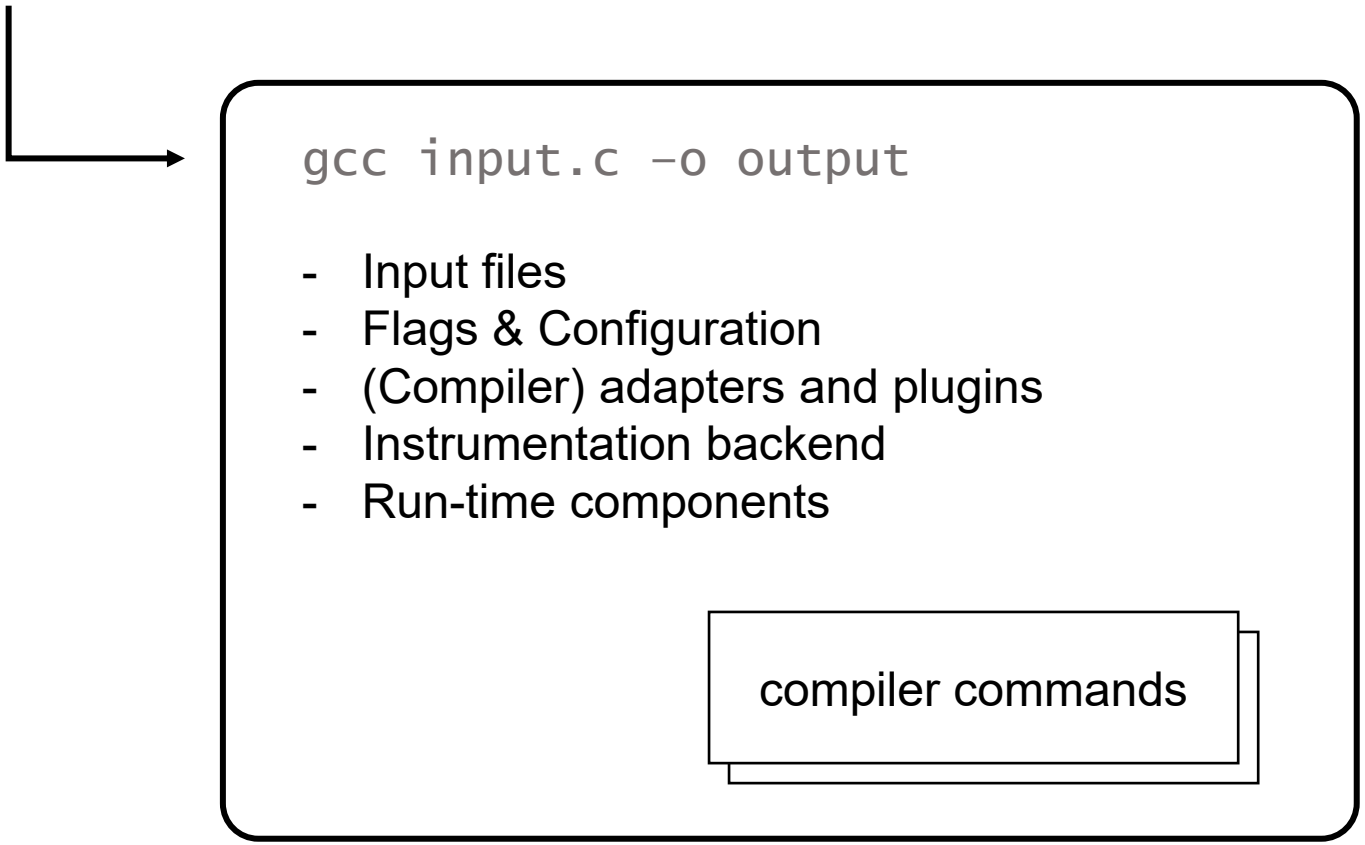
```
  void work(){
    measure_something();
    // ….....
    measure_something();
  }
```

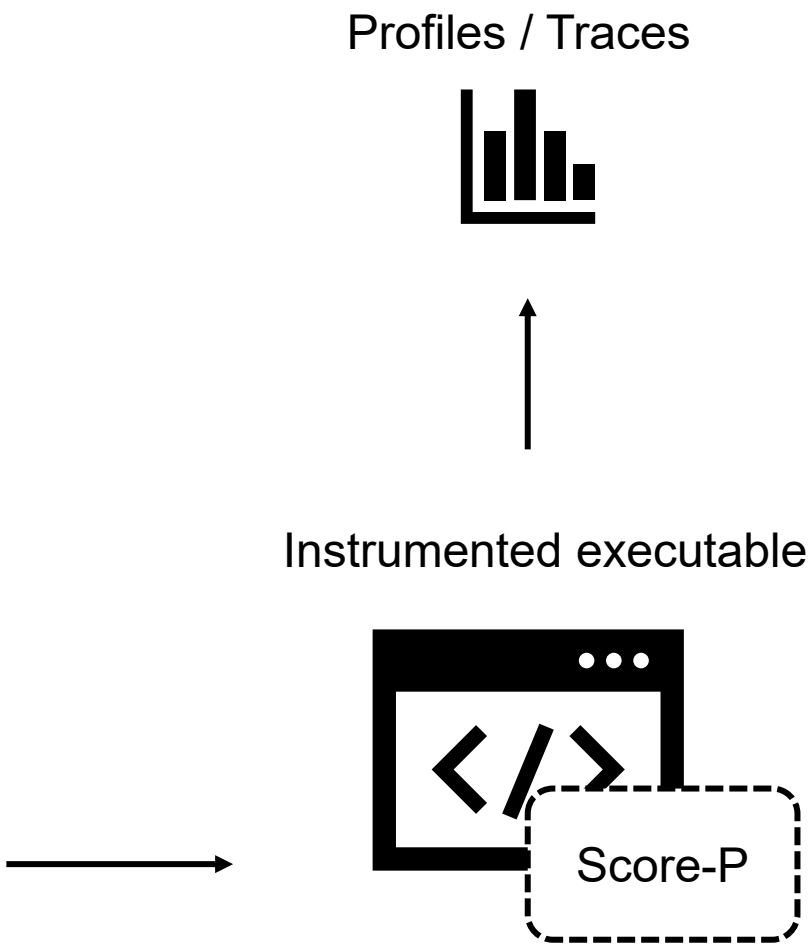
SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

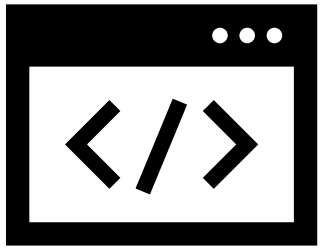
SCORE-P

```
scorep gcc input.c -o output
```

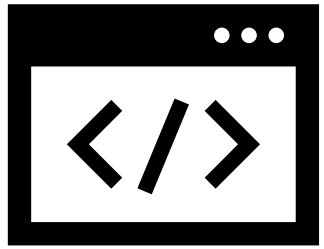




WHICH APPLICATION RUNS FASTER AND WHY?



test1



test2

Profile

type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
ALL	3,401,254,200	801,259,025	559.32	100.0	0.70	ALL
OMP	1,945,003,398	354,328,260	494.47	88.4	1.40	OMP
USR	1,449,170,840	445,898,713	31.60	5.6	0.07	USR
MPI	6,551,705	869,516	11.76	2.1	13.53	MPI
COM	528,216	162,528	21.49	3.8	132.24	COM
SCOREP	41	8	0.00	0.0	37.11	SCOREP

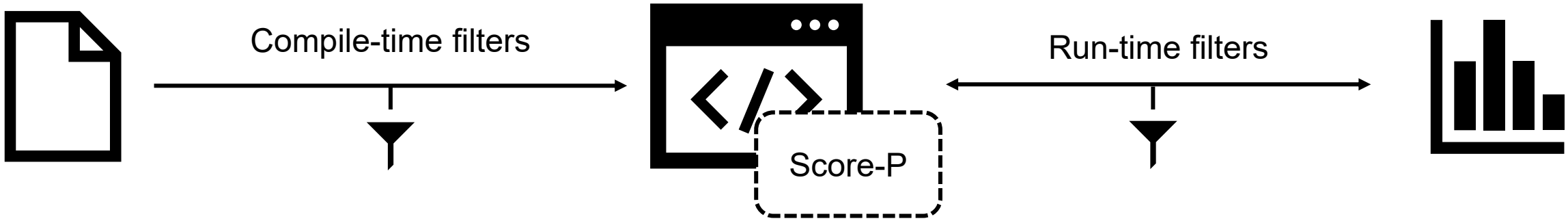
Problems

- Measurement overhead
- Disproportionality
- High runtime

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

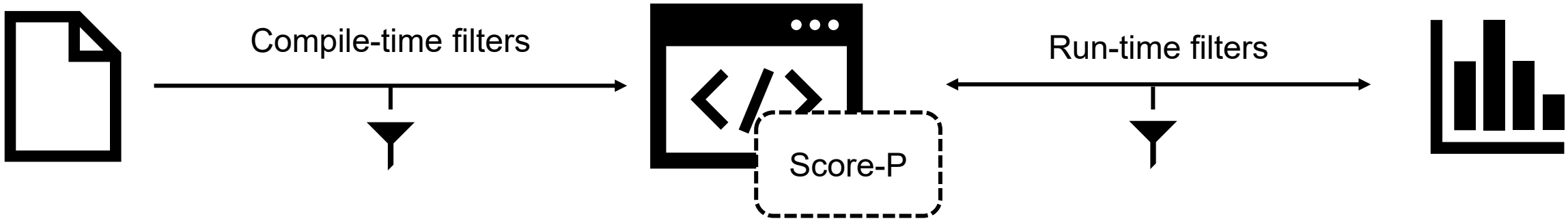
FILTERING WITH SCORE-P



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

FILTERING WITH SCORE-P



- Requires recompilation

- Residual overhead

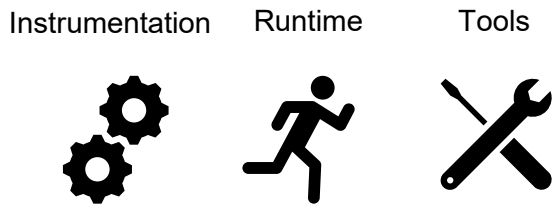
Static instrumentation

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

XRAY

- Function call tracing system
- Dynamic instrumentation
- Originally developed by Google
- Integrated into LLVM

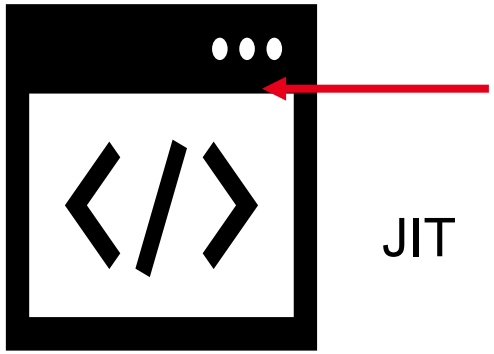


XRAY

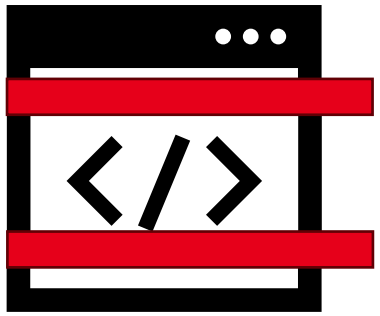
- Function call tracing system
- Dynamic instrumentation
- Originally developed by Google
- Integrated into LLVM

Instrumentation Runtime Tools

Fully dynamic instrumentation



XRay



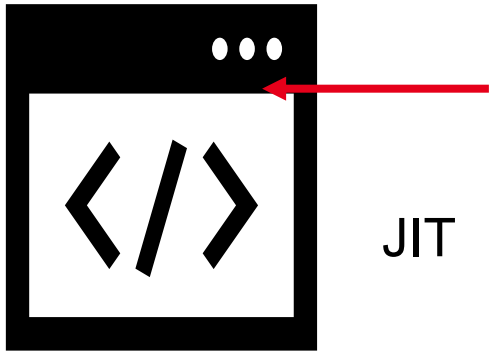
Sleds Patching Trampoline function

```
fun_entrysled:
 jmp . + 0x09
 (9 bytes of nops for x86)
 ...
 ... # fun code
fun_exitsled:
 retq
 (10 bytes of nops)
```

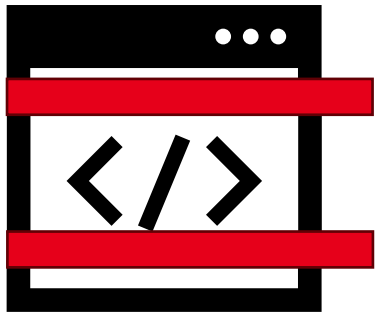
XRAY

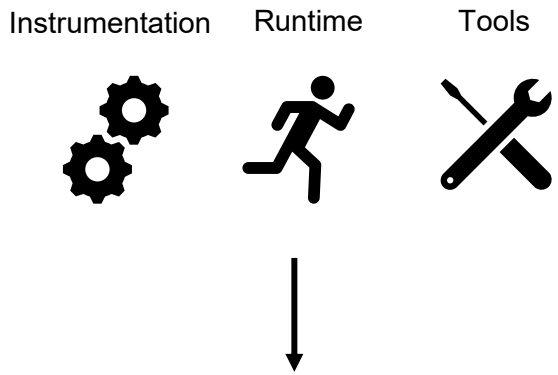
- Function call tracing system
- Dynamic instrumentation
- Originally developed by Google
- Integrated into LLVM

Fully dynamic instrumentation



XRay





Sleds — Patching — Trampoline function

```
fun_entrysled:
 mov $<function id>, %r10d
 call __xray_FunctionEntry
 ...
 ... # fun code
fun_exitsled:
 retq
 (10 bytes of nops)
```

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

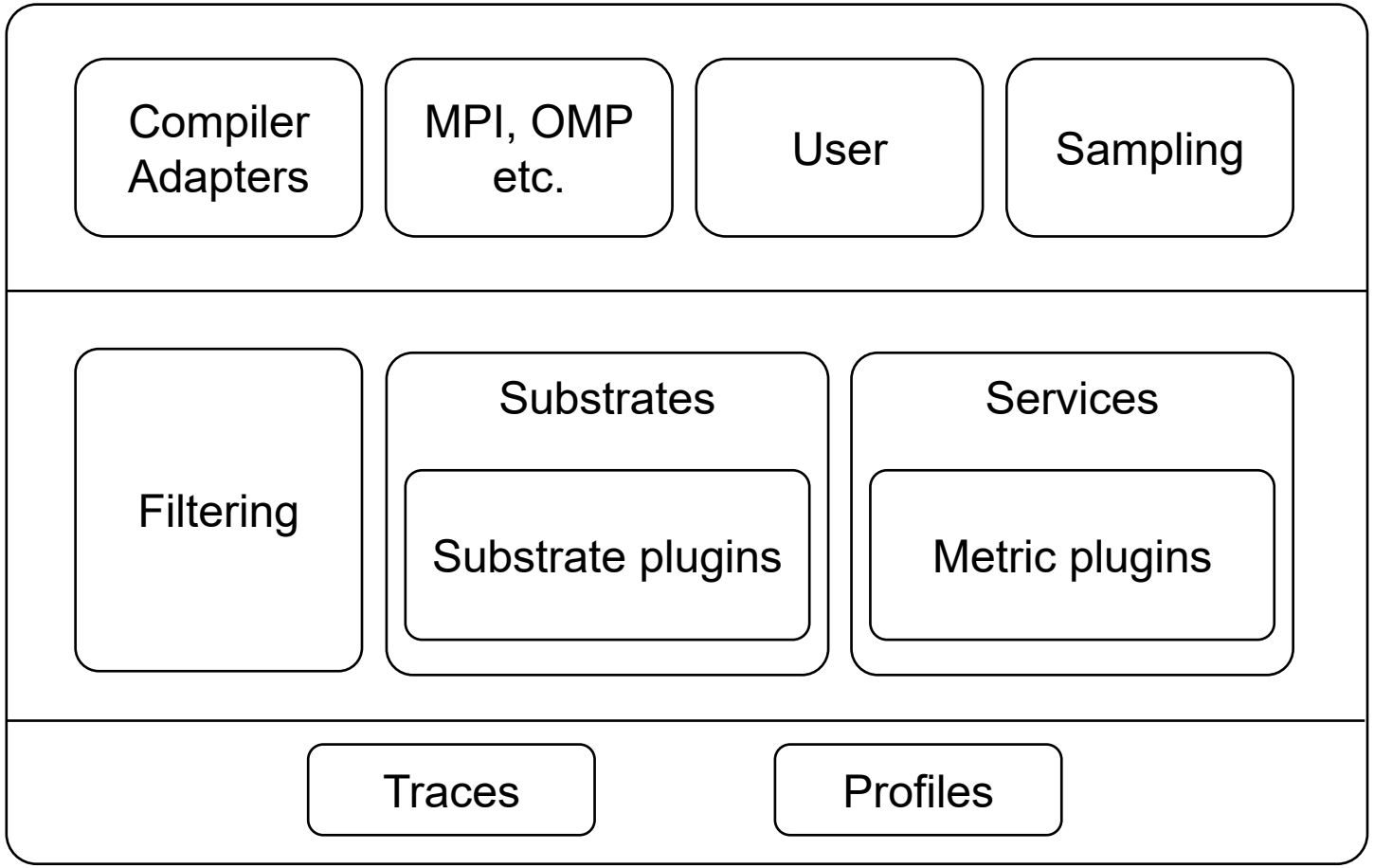
MOTIVATION

- Leverage XRay's capabilities
- Without leaving the comfort of Score-P
- Unify tool landscape
- Enable new workflows
- Performance improvements
- Reduce filtering overhead
- Remove need for recompilation

=> Extend Score-P to allow for usage of XRay as an instrumentation backend

IMPLEMENTATION

CONSIDERATIONS

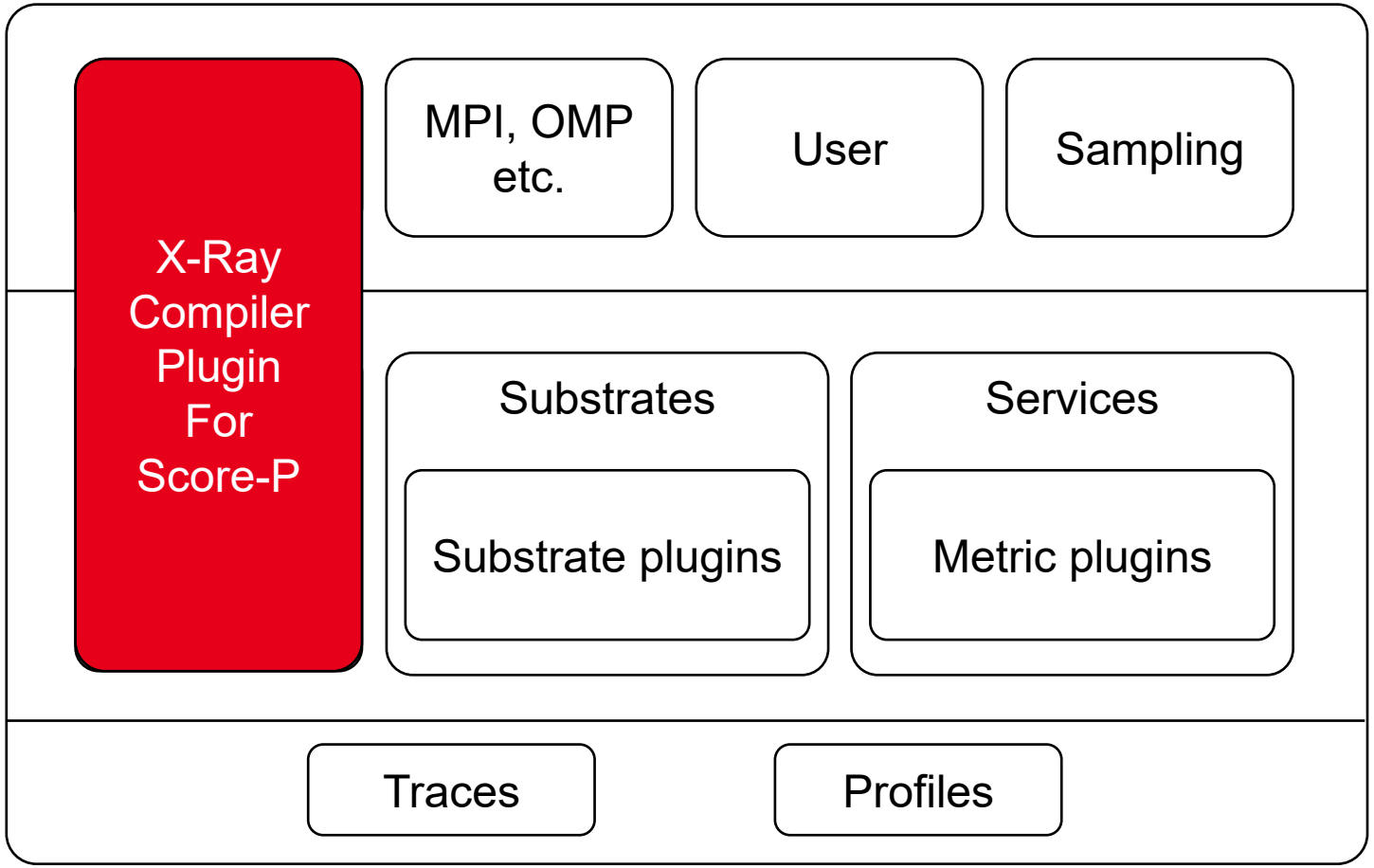


CONSIDERATIONS

Instrumentation

Run-time system

Results/Effects



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

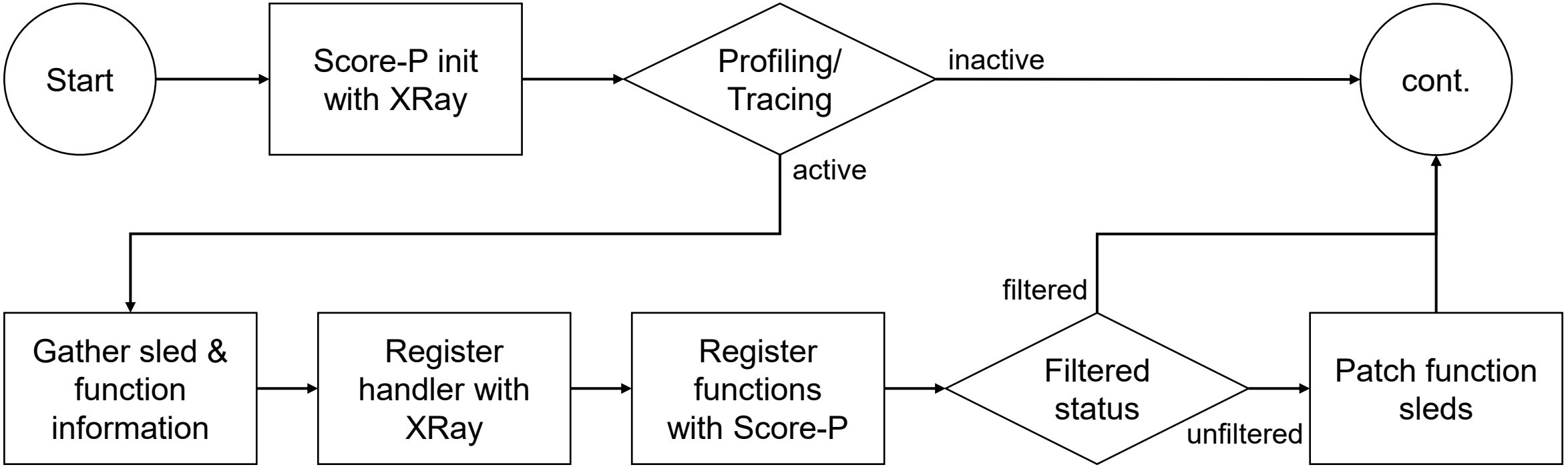
BUILD PROCESS

- XRay as compiler plugin
- Support determined during build
- No flags during runtime
- XRay components not linked if unwanted
- Currently mutually exclusive: LLVM vs. XRAY
- `--enable-xray-plugin` and `--disable-xray-plugin`

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

RUNTIME FLOWCHART



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

ADDITIONAL FEATURES

- Rudimentary compile time filter support with one-way translation
- XRay plugin specific flags
  - `--xray-instruction-threshold=<int>` to change the default xray instruction threshold
  - `--xray-plugin-arg=<str>` to pass arbitrary arguments the XRay plugin at compile time, arguments must be recognized by Xray
  - `--no-xray-compile-with-debug` to disable passing of a debug flag (mostly -g) during compilation, used to extract function names from the final executable
  - `--no-xray-delete-converted-filter` to keep temporary converted filter files instead of deleting them after compilation
- Help via `--help`
- Additional Score-P error codes

EVALUATION

SETUP

- GCC plugin (Score-P release 8.4)
- LLVM plugin commit a23c43d4 (Still in development)
- XRay plugin with no compile time filters, threshold 1
- Baselines with clang (llvm 18.1.8) and gcc (13.1.0)
- Performed on Lichtenberg II cluster stage 2 (node feature set i02)
  - 2 Intel® Xeon® Platinum 8470Q processors (104 cores total)
  - 512 GB of DDR5-4800 main memory
- Configurations:
  - Full profiling
  - Default filter
  - Fully filtered

```
scorep-score -g …/profile.cubex
```

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

BENCHMARK OVERVIEW

Benchmark suite / program	Language	Information	Benchmark suite	Variations
LULESH 2.0	C++	Livermore Unstructured Lagrange Explicit Shock Hydrodynamics	LLNL Proxy Apps	Serial
				OpenMP
				MPI
				OpenMP+MPI
433.milc	C	Gauge field theory	SPEC2006	
505.mcf_r	C	Scheduling	SPECrate2017 Integer	
508.namd_r	C++	Biomolecular simulation	SPECrate2017 Floating Point	
644.nab_s	C	Molecule simulation	SPECspeed2017 Floating Point	Serial
				OpenMP

INSTRUMENTATION THRESHOLD

- Xray heuristic: 200 instructions or function contains loops

Instruction threshold	Instrumented
1	254
3	254
5	254
10	252
50	62
100	55
200	46
500	42
1000	41

LULESH Debug build (793 total functions)

Instruction threshold	Instrumented
1	24
3	24
5	23
10	22
50	24
100	19
200	17
500	13
1000	12

LULESH Release Build (753 total functions)

- Better comparability: shadow LLVM compiler plugin

=> Default threshold of 1, filter at runtime

SCIENTIFIC COMPUTING
TECHNISCHE UNIVERSITÄT DARMSTADT

EXECUTABLE SIZE

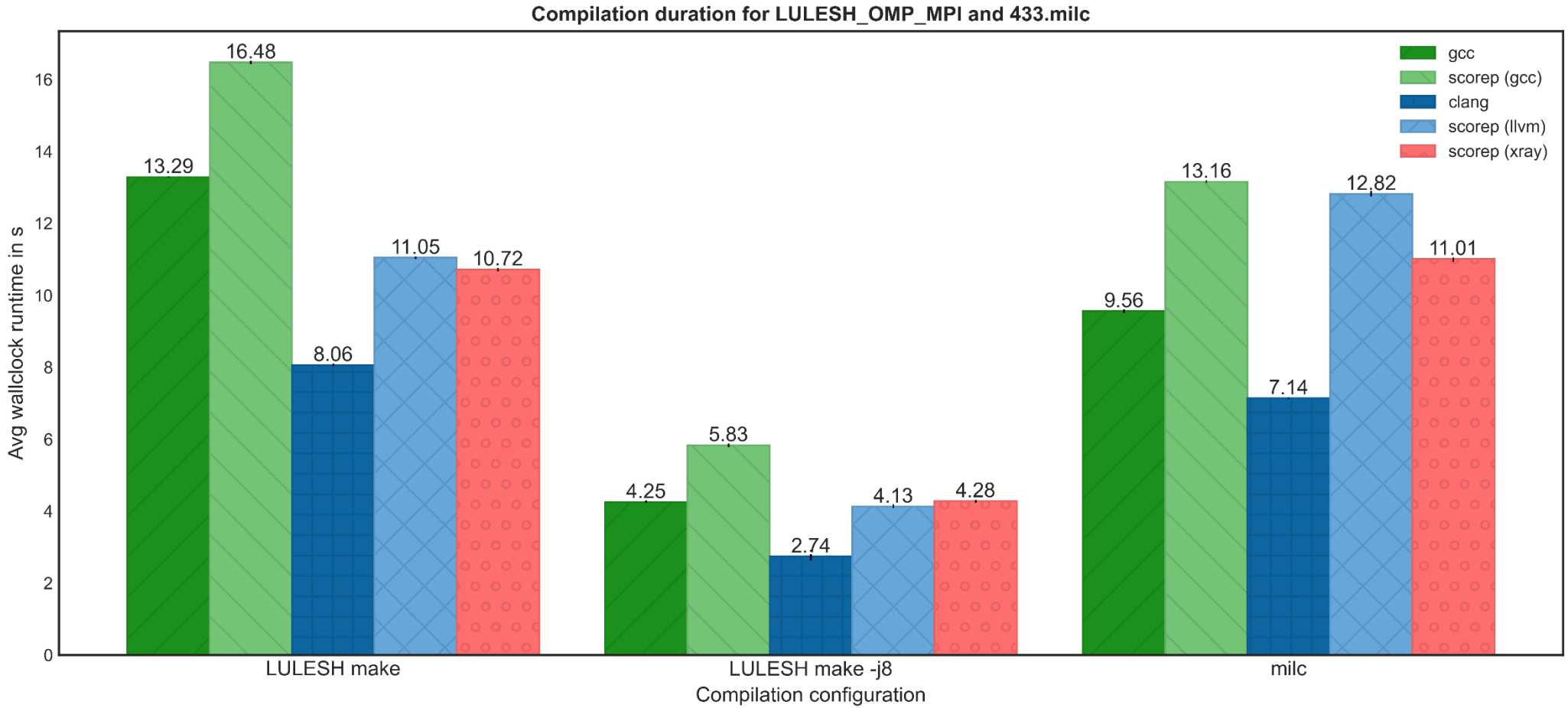
Application	Compilation configuration	Executable size
LULESH serial	clang	82 KiB
	gcc	90KiB
	Score-P (GCC)	115 KiB
	Score-P (LLVM)	335 KiB
	Score-P (XRay)	1212 KiB
LULESH OMP MPI	clang	151 KiB
	gcc	223KiB
	Score-P (GCC)	273 KiB
	Score-P (LLVM)	598 KiB
	Score-P (XRay)	1441 KiB
433.milc	clang	180 KiB
	gcc	183KiB
	Score-P (GCC)	255 KiB
	Score-P (LLVM)	711 KiB
	Score-P (XRay)	1364 KiB
644.nab_s OMP	clang	708 KiB
	gcc	973 KiB
	Score-P (GCC)	983 KiB
	Score-P (LLVM)	955 KiB
	Score-P (XRay)	1647 KiB

- ~1 MiB increase in size vs. base
- Mostly constant
- Hidden linkage in llvm/xray: static linking required
- Whole archive linking

SCIENTIFIC COMPUTING

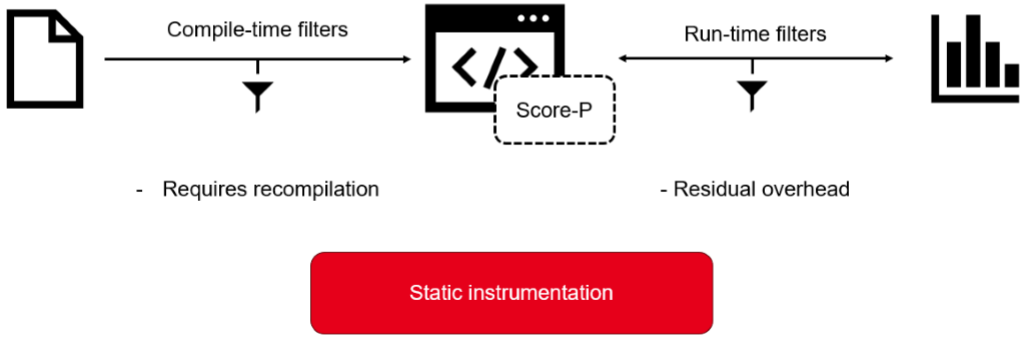
TECHNISCHE UNIVERSITÄT DARMSTADT

COMPILATION DURATION

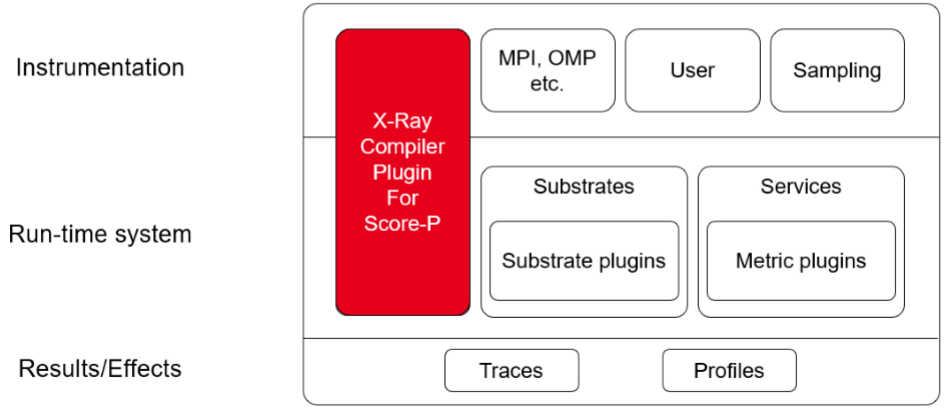


COMPILE- VS. RUNTIME FILTERING

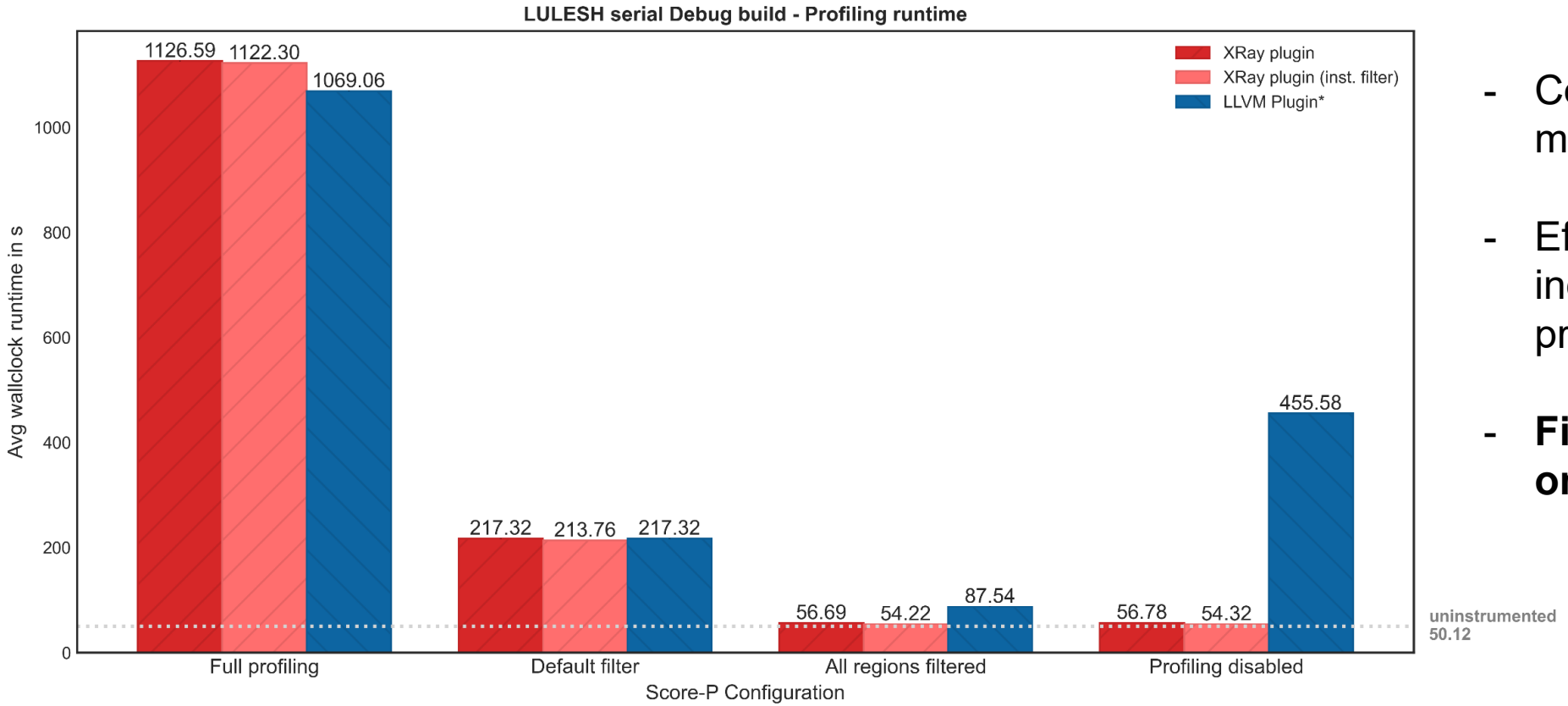
FILTERING WITH SCORE-P



CONSIDERATIONS



COMPILE- VS. RUNTIME FILTERING



- Compile time filtering is max. 4.33% faster
- Effect diminishes with increasing numbers of profiled regions
- **Filtering at runtime only is feasible**

SCIENTIFIC COMPUTING

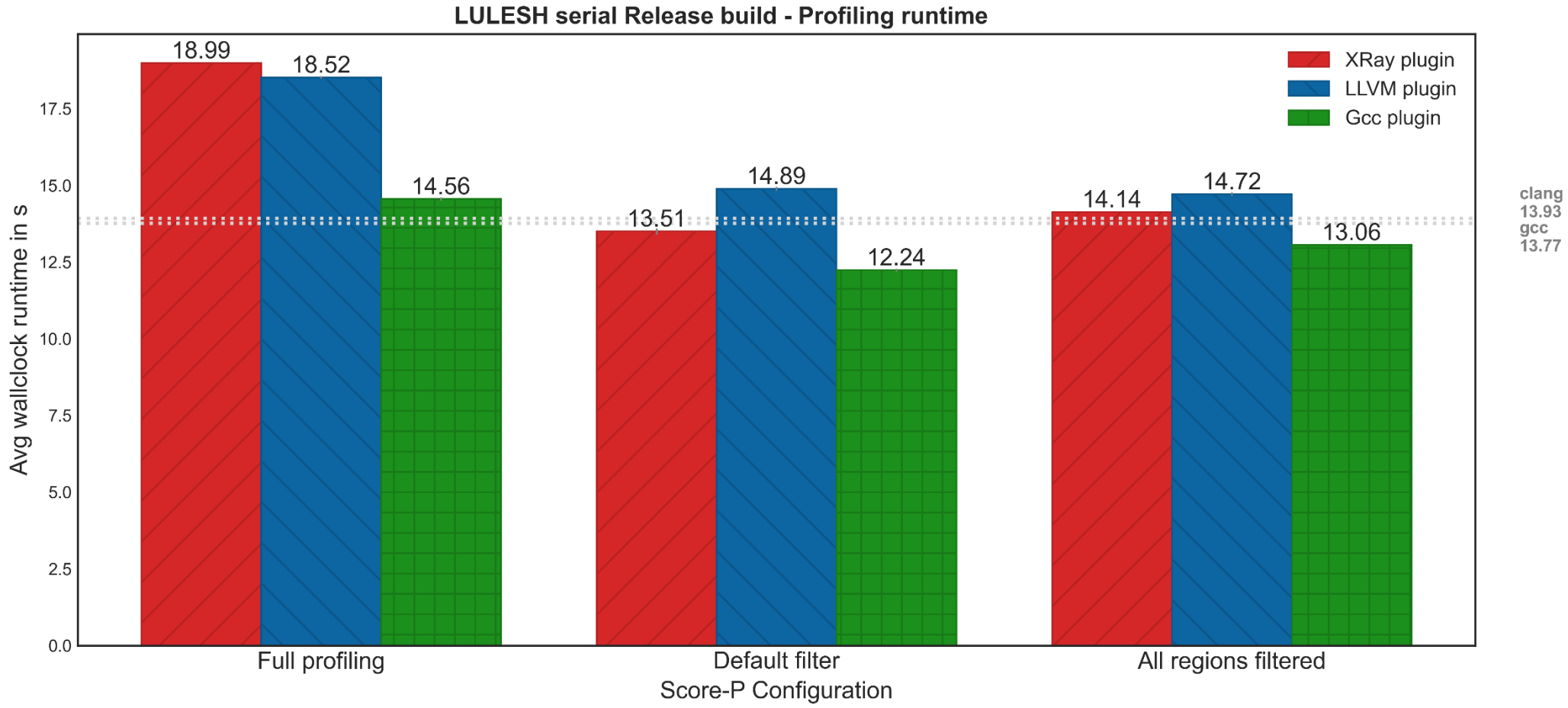
TECHNISCHE UNIVERSITÄT DARMSTADT

RUNTIME PERFORMANCE

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

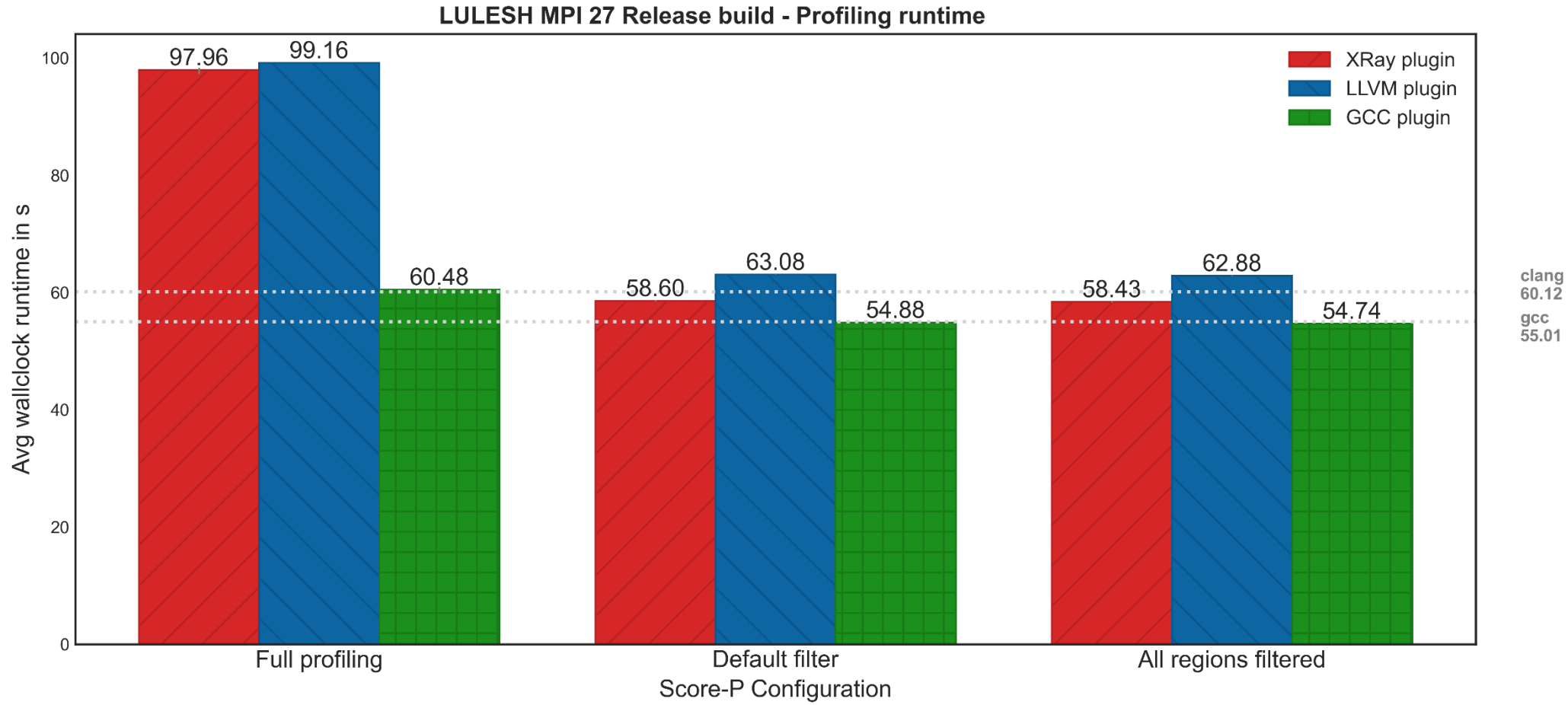
LULESH
SERIAL



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

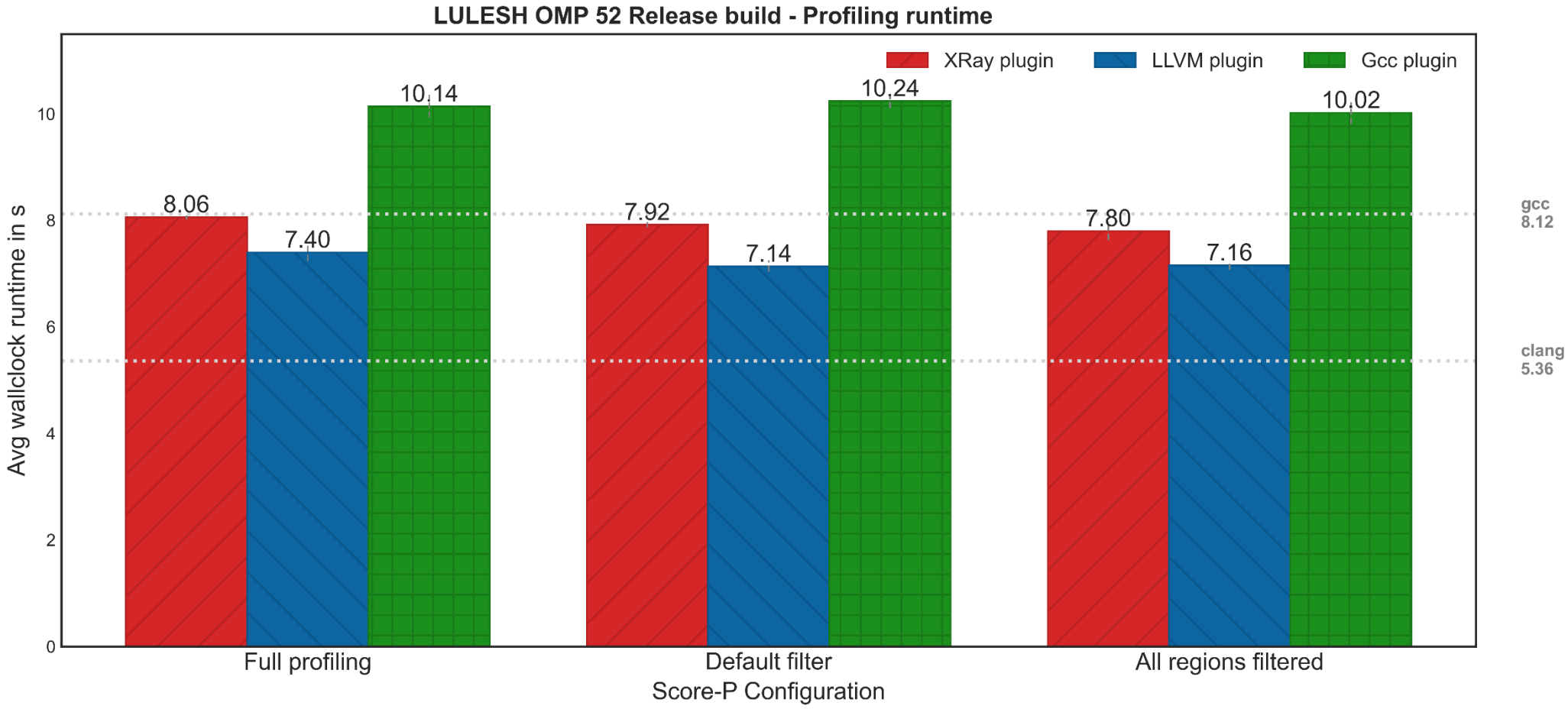
LULESH
MPI 27 PROCESSES



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

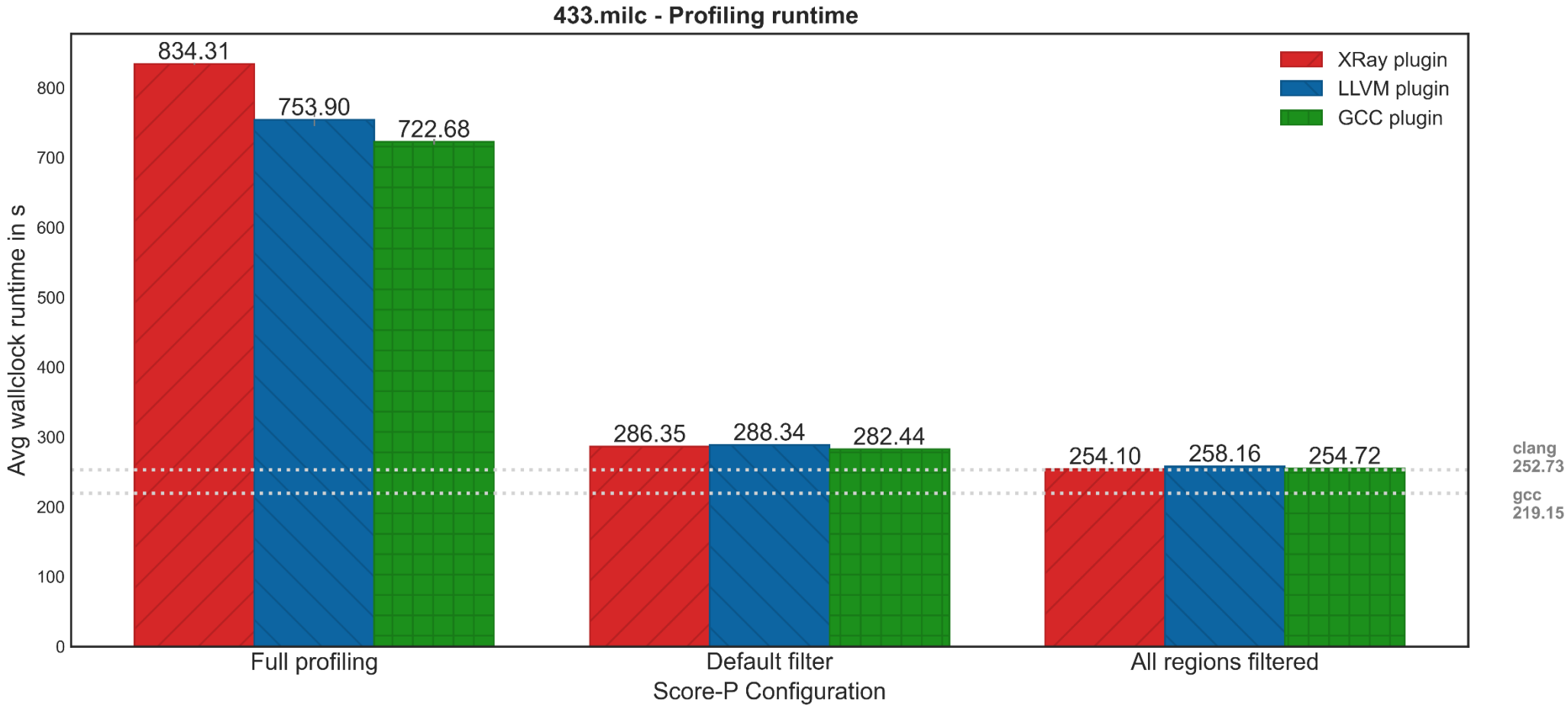
LULESH
OMP 52 THREADS



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

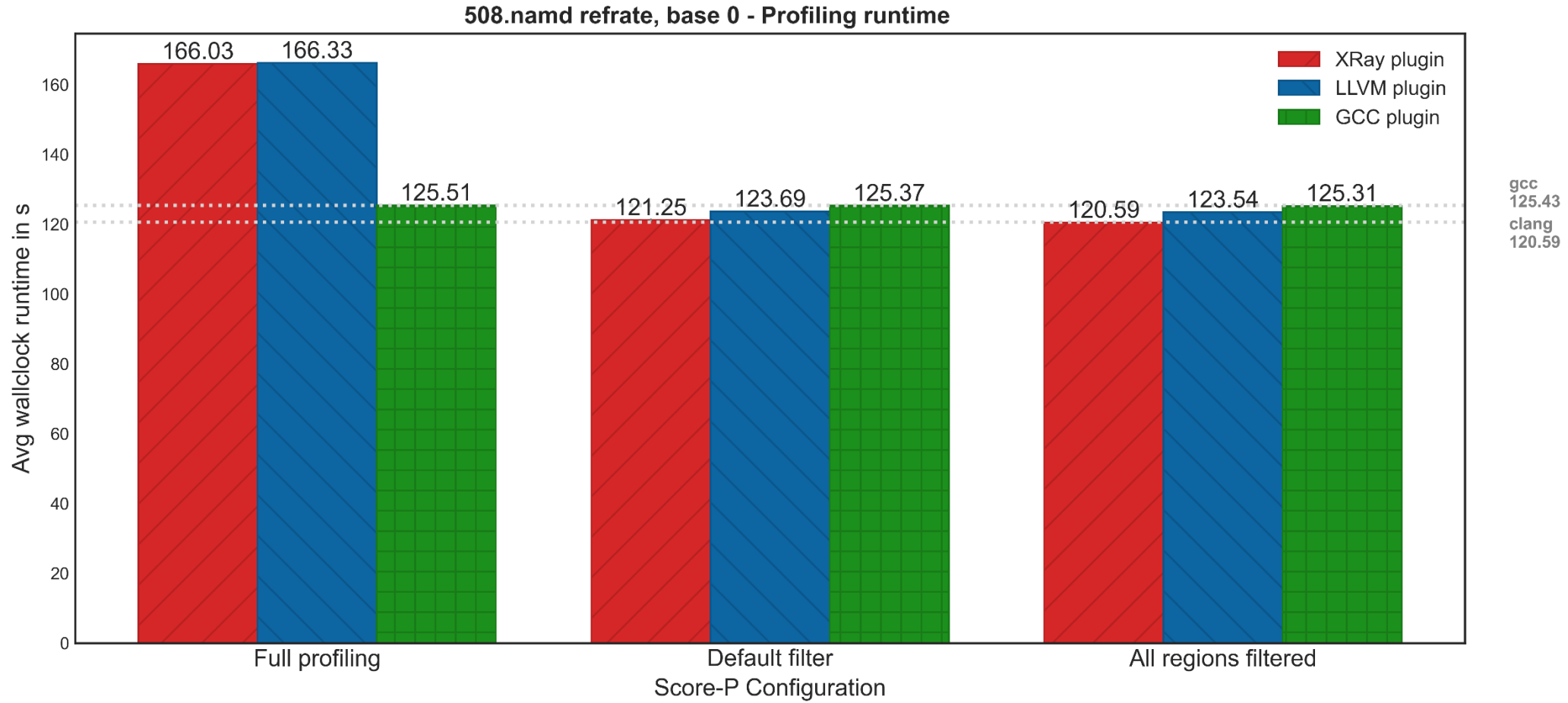
433.MILC



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

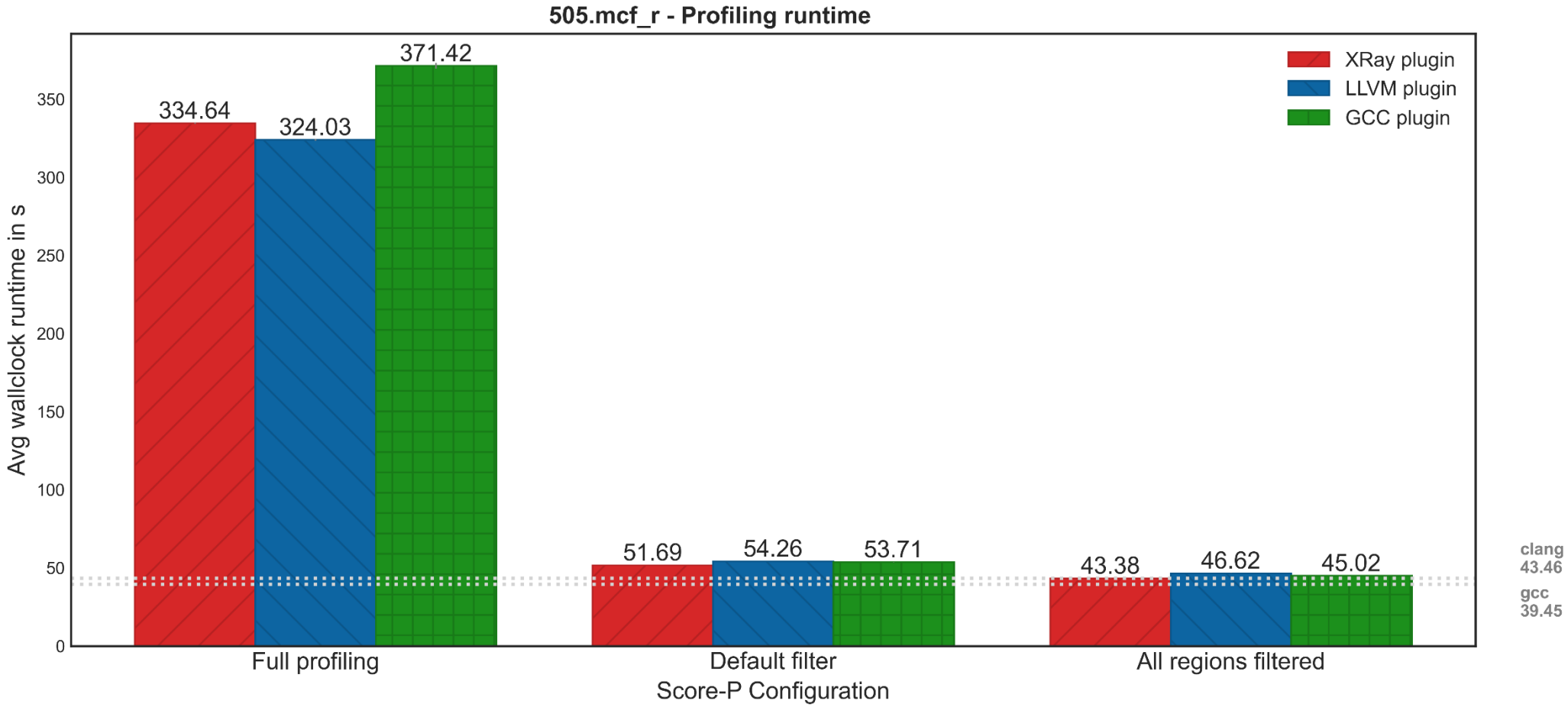
508.NAMD_R
(DATASET: REFRATE)



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

505.MCF_R
(DATASET: REFERENCE)



DISCUSSION

- Similar performance to LLVM plugin

- Increased executable size
- Limited compile-time filter support

- Reduced overhead for run-time filters
- Full runtime filtering feasible
- Faster filtering: Performance gains with number of filtered regions
- Negligible startup time

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

XRAY PLUGIN FOR SCORE-P

- Score-P with static instrumentation via compiler plugins
  - Re-compilation for compile time filtering is time consuming
  - Runtime filtering has remaining overhead
- LLVM Xray for partially dynamic instrumentation via sleds
- XRay support in Score-P via a compiler plugin (adapter)
- Patching (Filtering) performed at runtime, with compile time support
- Evaluation across different C/C++ programs
- Comparable measurement overhead for full instrumentation
- Significantly reduced overhead for filtered regions

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

FUTURE WORK

- Eliminate current shortcomings (Executable size, indirection)
- User-facing API for safe (un)patching
- Filter conversions
- Extend Xray
- Shared library support

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

THANK YOU!

Feel free to ask questions!

APPENDIX

IMPLEMENTATION

API

```
namespace XRayPlugin {
  static std::vector<scorep_compiler_region_description> *regions;

  static scorep_compiler_region_description createRegionDesc(
    std::string &funcNameMangled, std::string &funcNameDemangled, std::string &sourceFile,
    const uint32_t   startLine, const uint32_t endLine) XRAY_INSTRUMENT_NEVER;

  static bool buildRegionsForExecutable(std::string &execFileName) XRAY_INSTRUMENT_NEVER;
  static bool registerAndPatch() XRAY_INSTRUMENT_NEVER;
  static inline std::vector copyRegionHandles(std::vector *regiondescs) XRAY_INSTRUMENT_NEVER;
  static void handleInstrumentationPoint(int32_t fid, XRayEntryType entryType) XRAY_INSTRUMENT_NEVER;
  static inline bool shouldInitXray() XRAY_INSTRUMENT_NEVER;
  static SCOREP_ErrorCode initXRay() XRAY_INSTRUMENT_NEVER;
  static void cleanupXRay() XRAY_INSTRUMENT_NEVER;
}
```

20.09.2024 Scientific Computing | Paul Adelmann | 15th International Parallel Tools Workshop Dresden 41

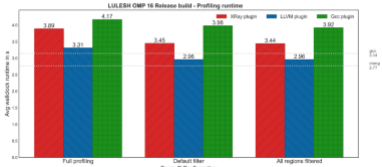
RELATED WORK

- Score-P LLVM plug-in
- Dynamic instrumentation tools such as DynamoRIO, Pin, DynInst, DynCaPI
- Usages for instrumentation: code coverage etc
- Valgrind, TypeART
- Instrumentation in other languages
- Score-P for Python
- Security context

20.09.2024 Scientific Computing | Paul Adelmann | 15th International Parallel Tools Workshop Dresden 40

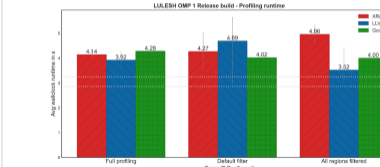
RUNTIME EVALUATION

LULESH
OMP 16 THREADS



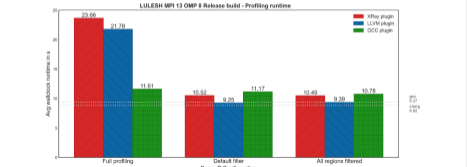
RUNTIME EVALUATION

LULESH
OMP 1 THREAD



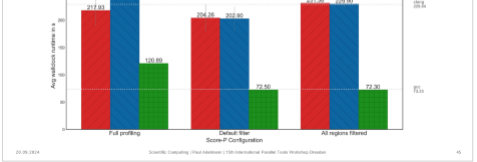
RUNTIME EVALUATION

LULESH MPI+OPENMP
(13 PROCESSES WITH 8 THREADS EACH)



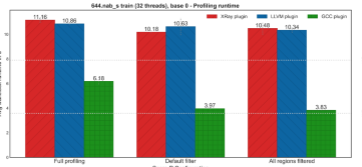
RUNTIME EVALUATION

644.NAB_S SERIAL
(DATASET: TRAIN)



EVALUATION

644.NAB_S
OPENMP 32 THREADS (DATASET: TRAIN)



RELATED WORK

- Score-P LLVM plug-in
- Dynamic instrumentation tools such as DynamoRIO, Pin, DynInst, DynCaPI
- Usages for instrumentation: code coverage etc
- Valgrind, TypeART
- Instrumentation in other languages
- Score-P for Python
- Security context

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

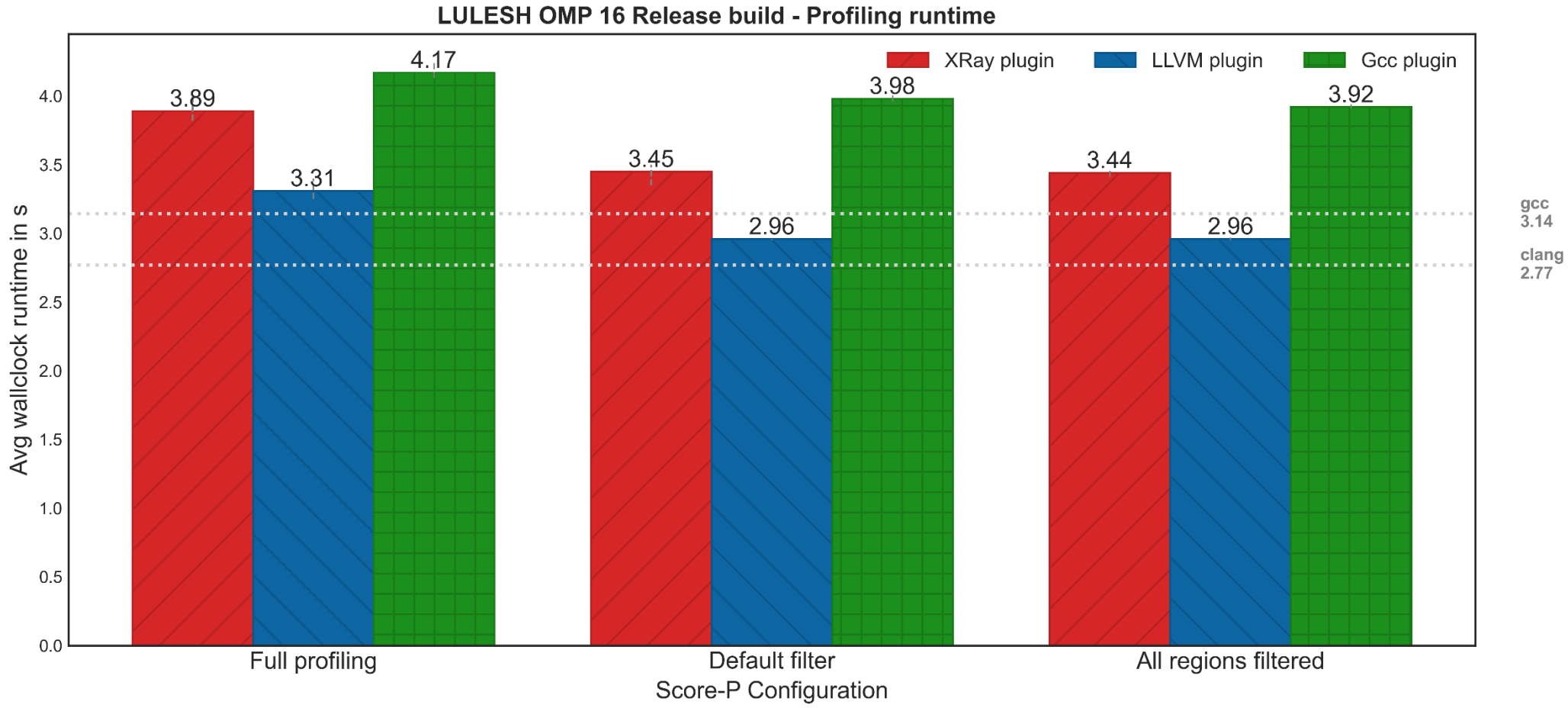
API

```
namespace XRayPlugin {
  static std::vector<scorep_compiler_region_description> *regions;

  static scorep_compiler_region_description createRegionDesc(
    std::string &funcNameMangled, std::string &funcNameDemangled, std::string &sourceFile,
    const uint32_t   startLine, const uint32_t endLine) XRAY_INSTRUMENT_NEVER;

  static bool buildRegionsForExecutable(std::string &execFileName) XRAY_INSTRUMENT_NEVER;

  static bool registerAndPatch() XRAY_INSTRUMENT_NEVER;

  static inline std::vector copyRegionHandles(std::vector *regiondescs) XRAY_INSTRUMENT_NEVER;

  static void handleInstrumentationPoint(int32_t fid, XRayEntryType entryType) XRAY_INSTRUMENT_NEVER;

  static inline bool shouldInitXray() XRAY_INSTRUMENT_NEVER;

  static SCOREP_ErrorCode initXRay() XRAY_INSTRUMENT_NEVER;

  static void cleanupXRay() XRAY_INSTRUMENT_NEVER;
}
```

SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

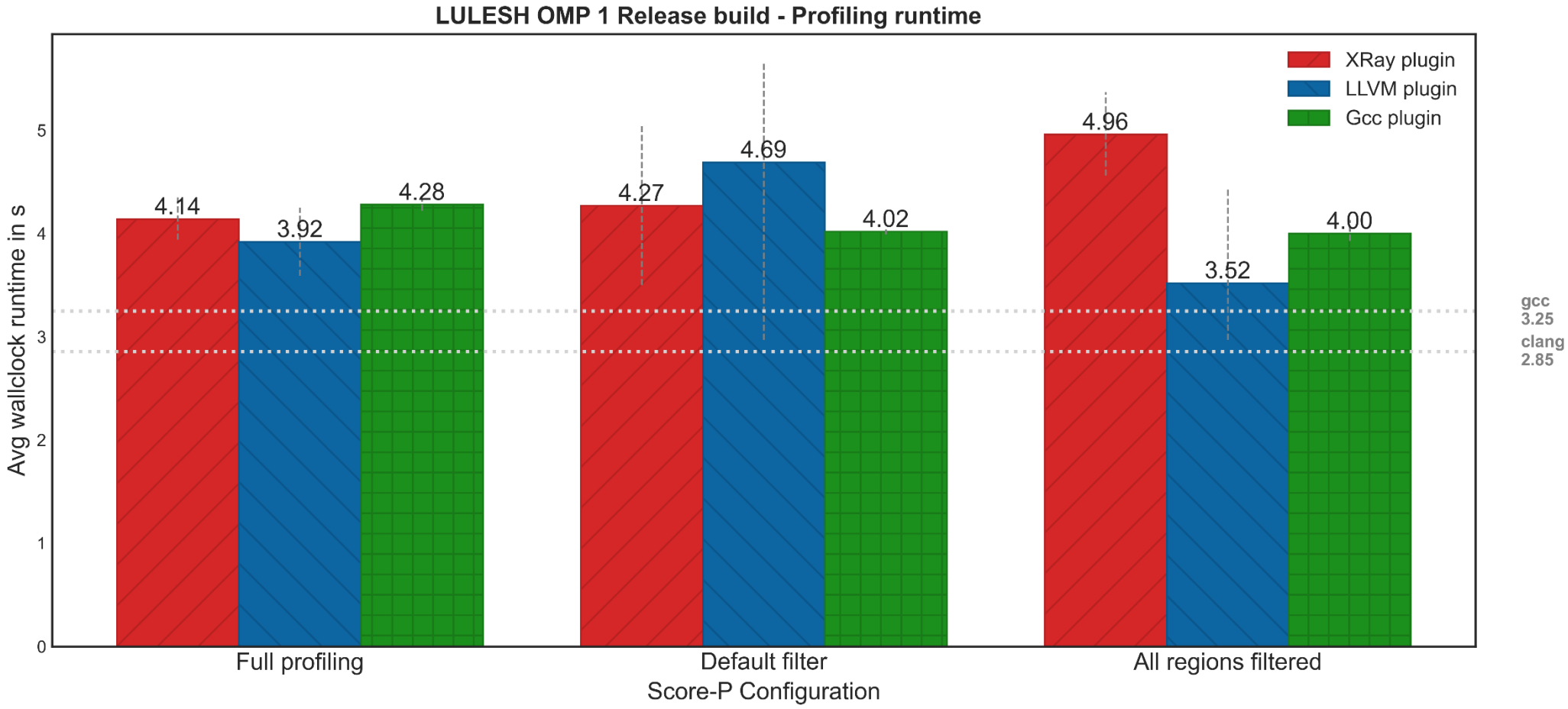
LULESH
OMP 16 THREADS



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

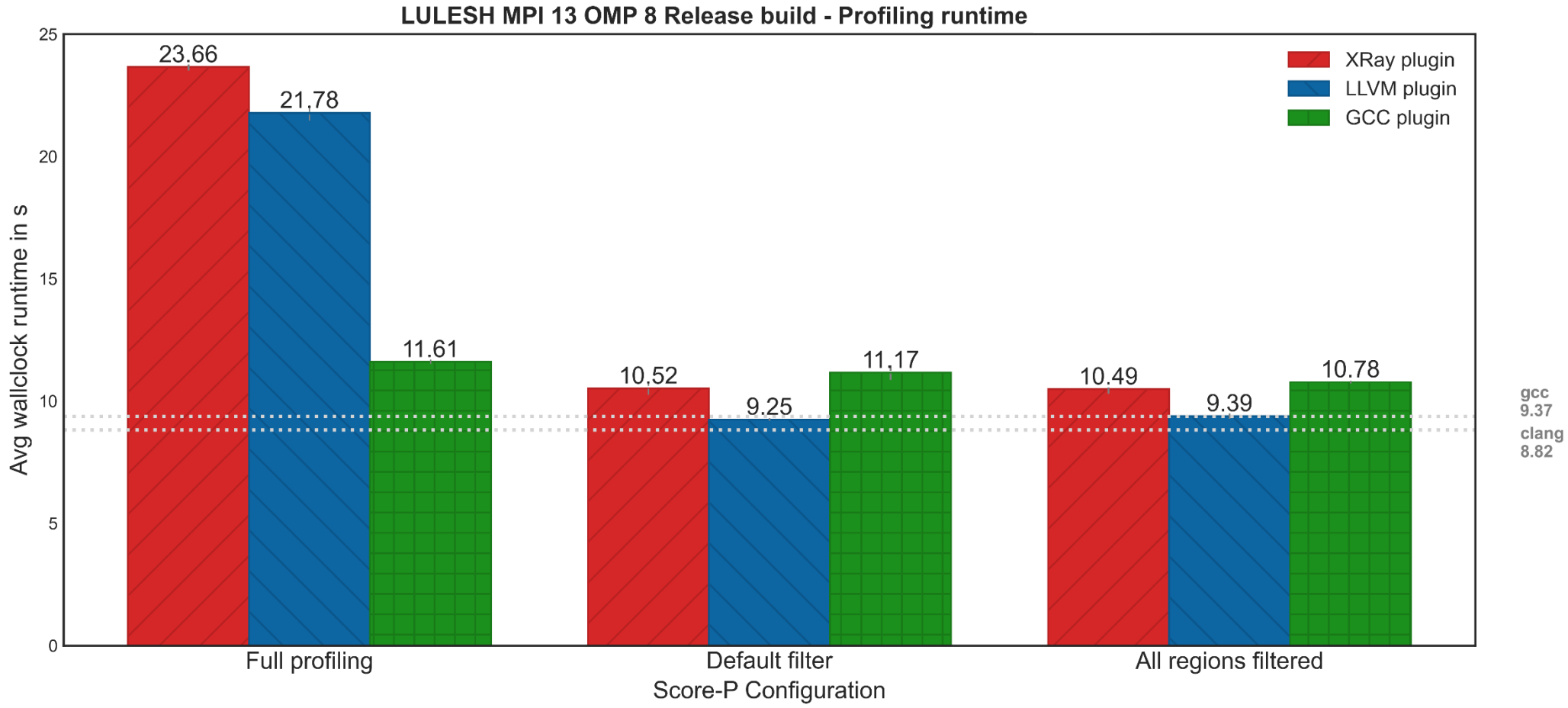
LULESH
OMP 1 THREAD



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

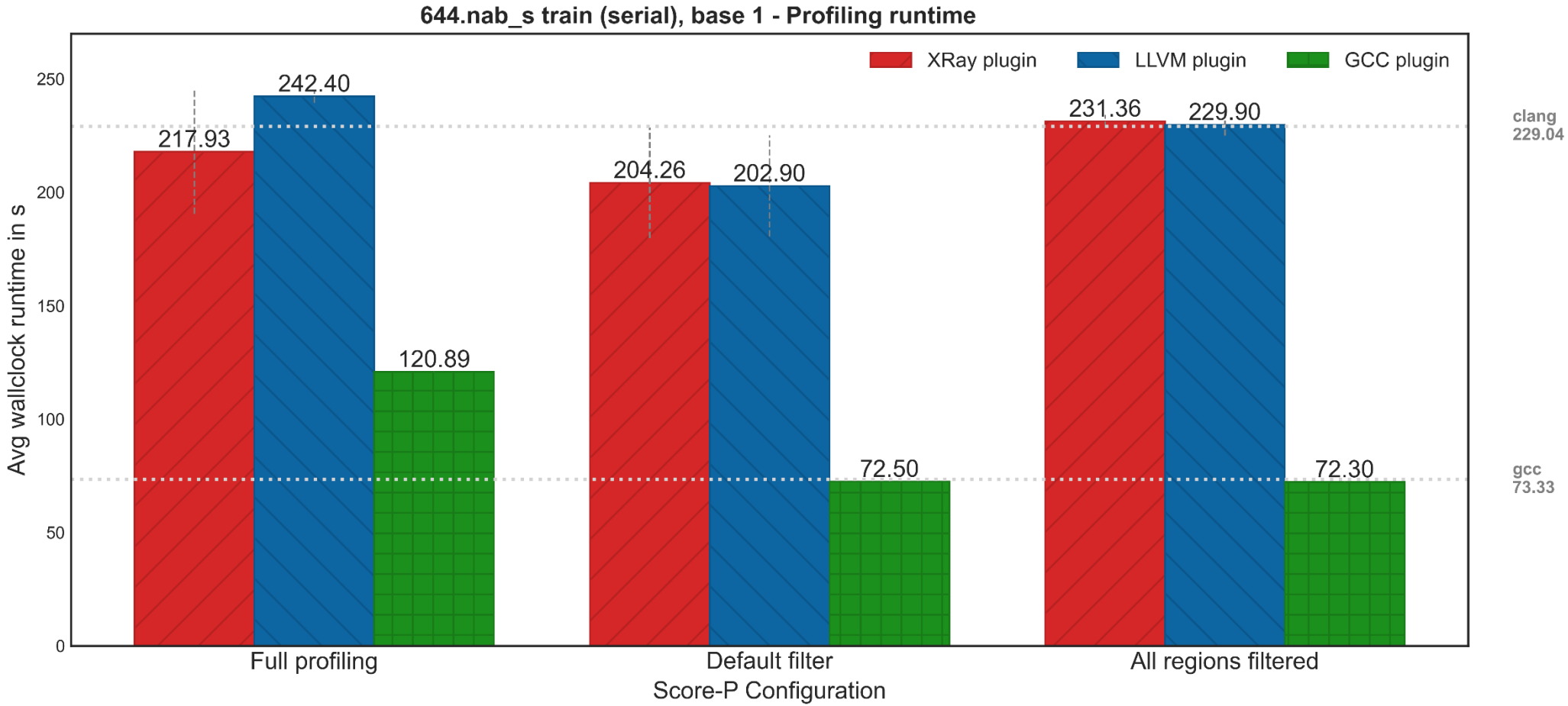
LULESH MPI+OPENMP
(13 PROCESSES WITH 8 THREADS EACH)



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

644.NAB_S SERIAL (DATASET: TRAIN)



SCIENTIFIC COMPUTING

TECHNISCHE UNIVERSITÄT DARMSTADT

644.NAB_S
OPENMP 32 THREADS (DATASET: TRAIN)