INTEGRATING XRAY INTO THE HPC TOOL ECOSYSTEM

EuroLLVM'25 Berlin

Sebastian Kreutzer
Scientific Computing, TU Darmstadt
sebastian.kreutzer@tu-darmstadt.de

Christian Bischof
Scientific Computing, TU Darmstadt
christian.bischof@tu-darmstadt.de

COMING UP

- Short introduction to XRay
- How XRay can benefit HPC performance tools
- Ongoing and future work towards better integration

BACKGROUND

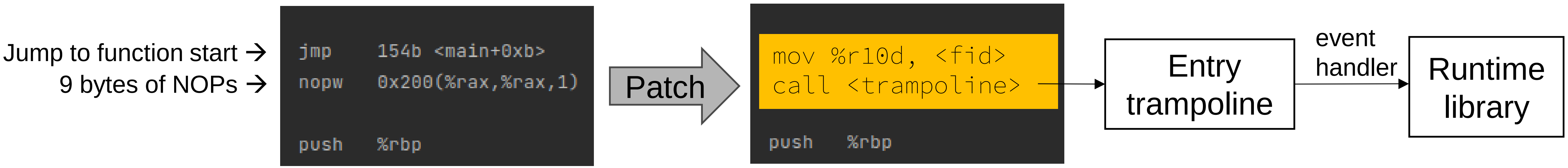
- XRay is LLVM's integrated instrumentation and tracing solution, consisting of
  - A hybrid instrumentation approach
  - A runtime library with built-in tracing modes
  - Tools to convert and analyze the collected data
- Originally developed by Google for performance debugging of applications running in production
  - Compile with `-fxray-instrument` to insert patchable instrumentation points
  - To start tracing, call `__xray_patch()`
  - Afterwards, convert and analyze the trace with the `llvm-xray` utility
- Talk by Dean Berris: https://llvm.org/devmtg/2017-10/slides/Berris_XRay_in_LLVM.pdf

BACKGROUND

- XRay is LLVM's integrated instrumentation and tracing solution, consisting of
  - **A hybrid instrumentation approach**
  - A runtime library with built-in tracing modes
  - Tools to convert and analyze the collected data
- Originally developed by Google for performance debugging of applications running in production
  - Compile with `-fxray-instrument` to insert patchable instrumentation points
  - To start tracing, call `__xray_patch()`
  - Afterwards, convert and analyze the trace with the `llvm-xray` utility
- Talk by Dean Berris: https://llvm.org/devmtg/2017-10/slides/Berris_XRay_in_LLVM.pdf

XRAY INSTRUMENTATION

- Hybrid approach:
  - Compile-time: Insert “NOP Sleds”
  - Runtime: Replace NOPs with call to profiling handler



Jump to function start →
9 bytes of NOPs →

```
jmp     154b <main+0xb>
nopw    0x200(%rax,%rax,1)

push    %rbp
```

Patch

```
mov %r10d, <fid>
call <trampoline>

push    %rbp
```

Entry trampoline

event handler

Runtime library

Near-zero overhead when unpatched → Single binary usable for profiling and production

No binary re-ordering/JIT required → Less invasive than fully dynamic instrumentation

Fast, thread-safe patching → adjustments possible at arbitrary points at runtime

INSTRUMENTATION IN HPC TOOLS

Tool:

Score-P

TAU

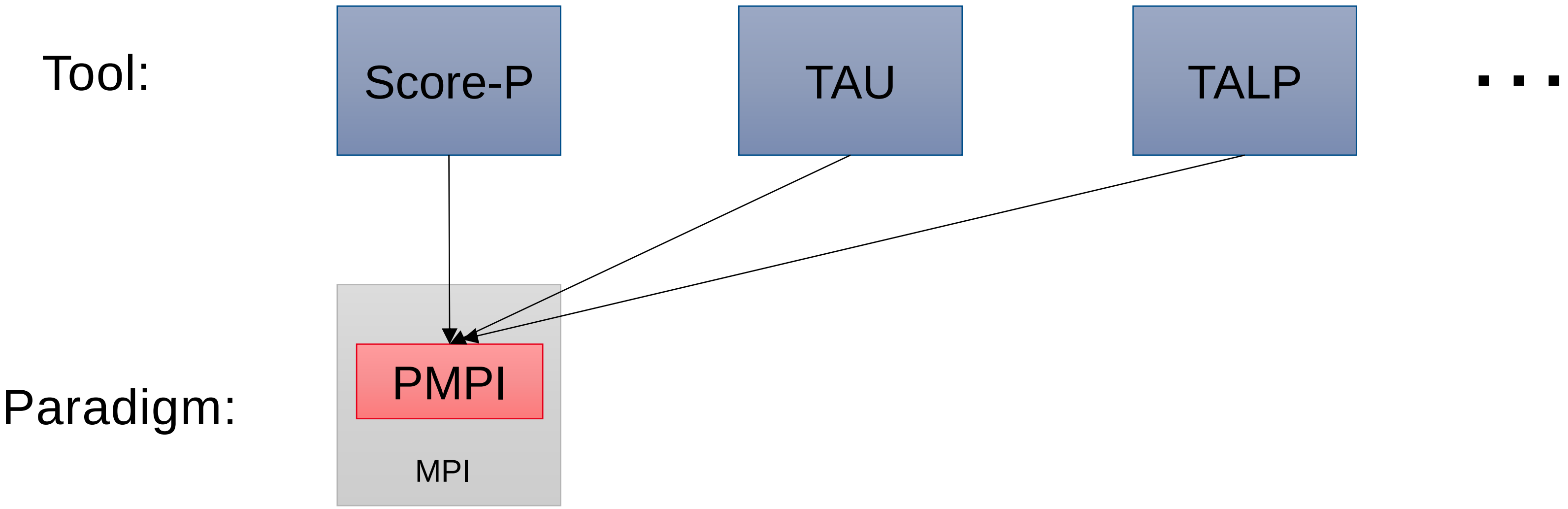
TALP

. . .

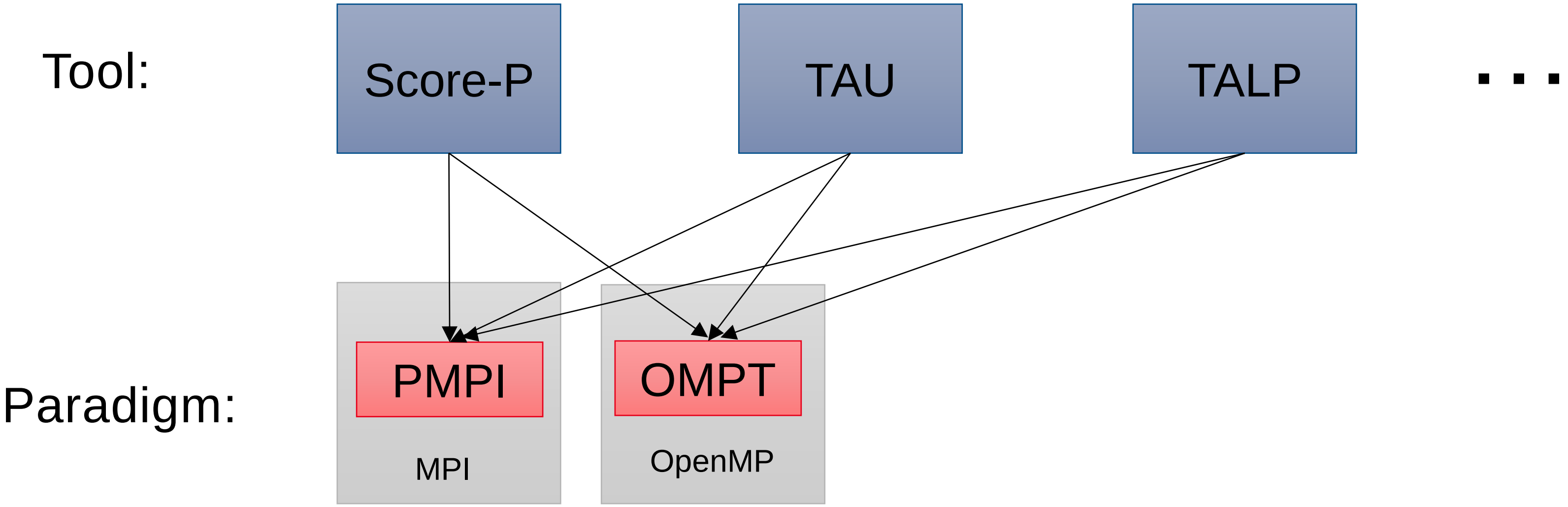
- Instrumentation-based tools
- Support for different programming models
  - MPI, OpenMP, CUDA, SHMEM…

Score-P: https://www.vi-hps.org/projects/score-p/
TAU: https://www.cs.uoregon.edu/research/tau/home.php
TALP: https://pm.bsc.es/ftp/dlb/doc/user-guide/how_to_run_talp.html

INSTRUMENTATION IN HPC TOOLS

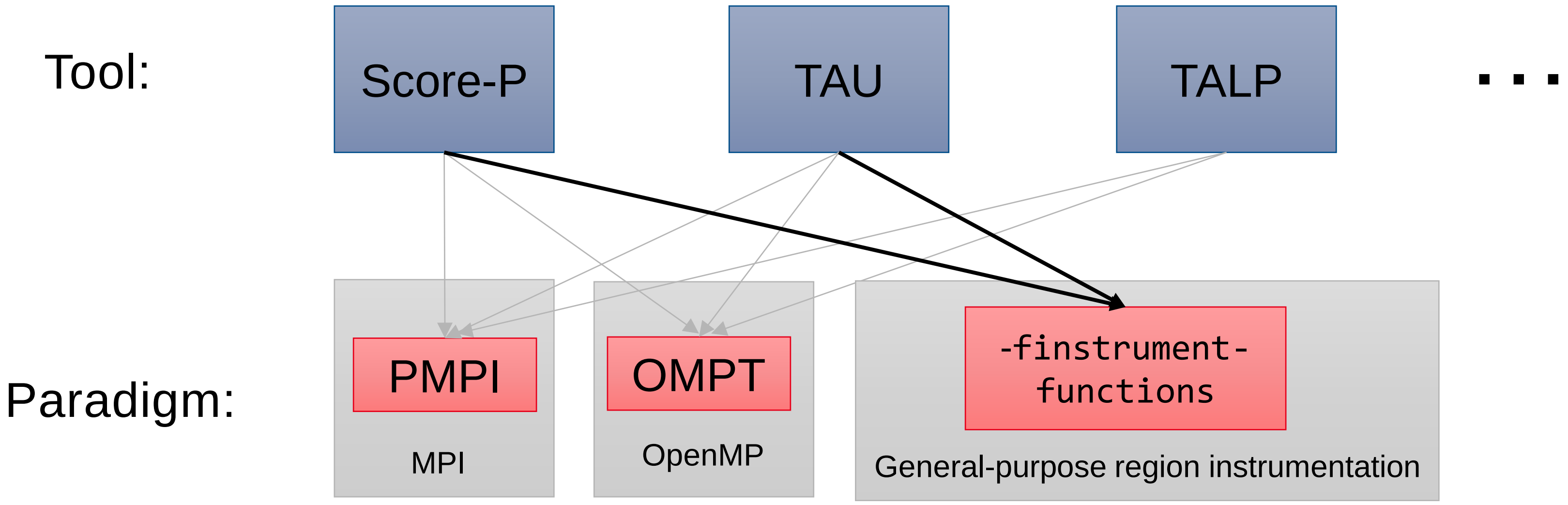


INSTRUMENTATION IN HPC TOOLS



→ (Semi-)standardized support for common parallel programming models

INSTRUMENTATION IN HPC TOOLS

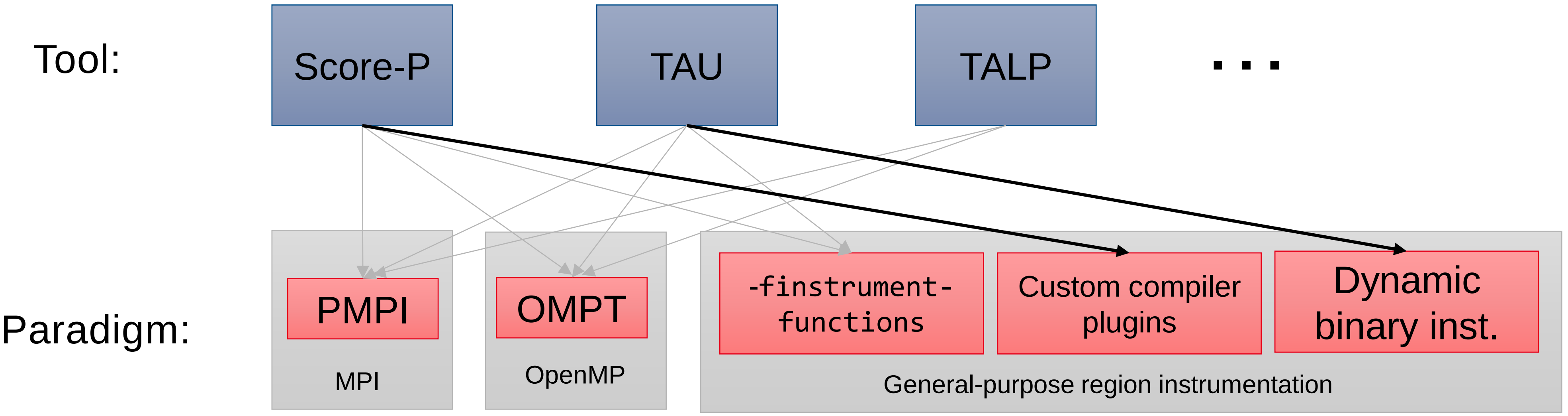


+ Supported by most compilers
- Limited selection control
- Basic interface
- High overhead

Example:

```
void kernel(double* A, int n) {
    __cyg_profile_func_enter(&kernel, ...)
    for (int i = 0; i < n; i++) {
        …
    }
    __cyg_profile_func_exit(&kernel, ...)
}
```

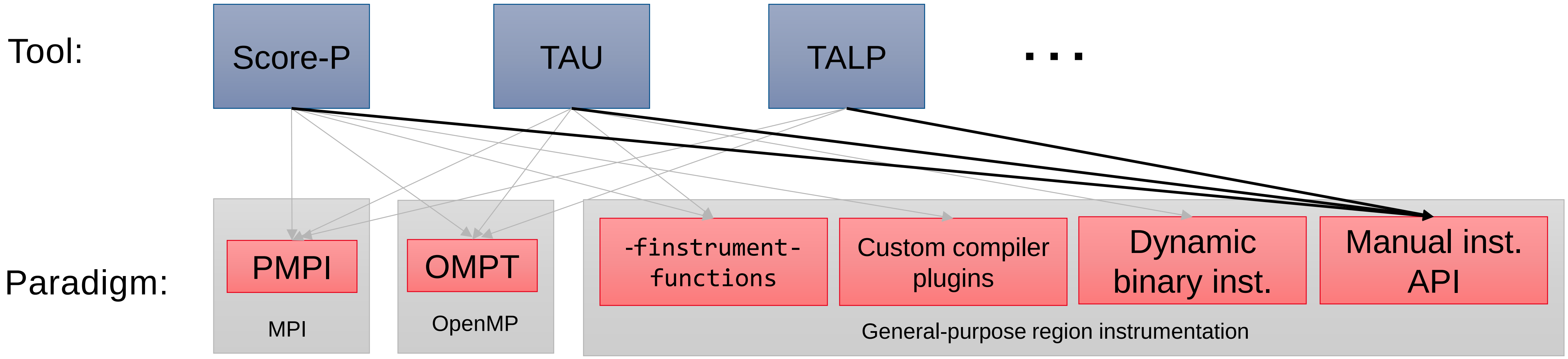
INSTRUMENTATION IN HPC TOOLS



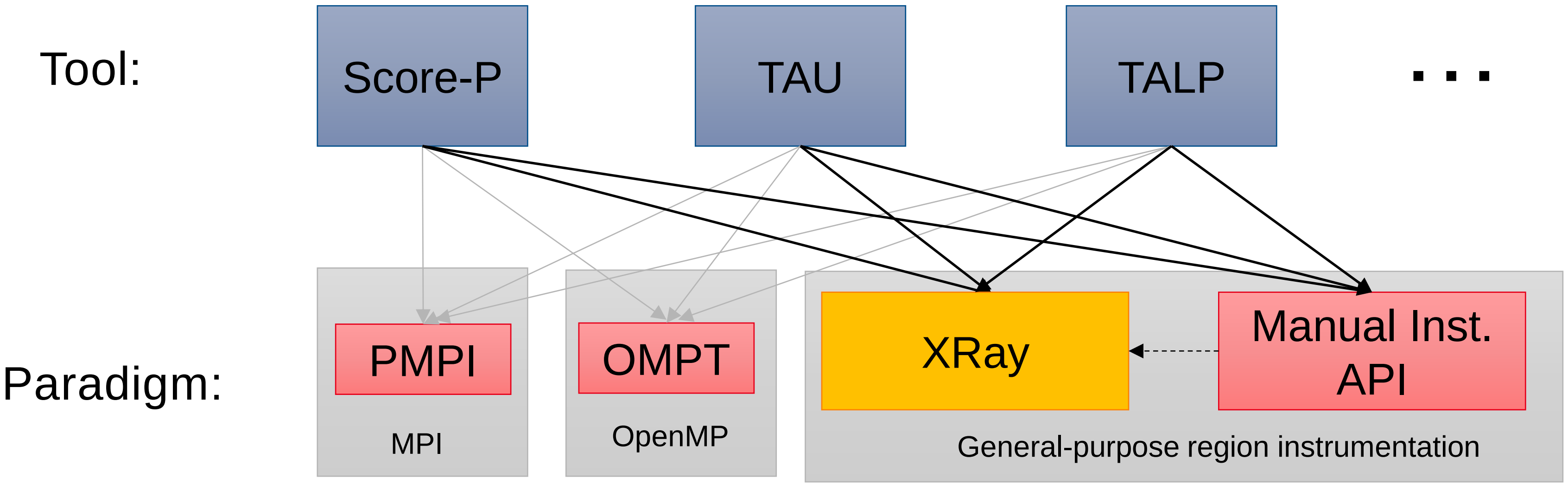
+ Mitigate `-finstrument-functions` issues

- Development/maintenance cost

INSTRUMENTATION IN HPC TOOLS



INSTRUMENTATION IN HPC TOOLS



?

+ Fast, dynamic adjustment
+ Low overhead
+ Integration effort similar to `-finstrument-functions`

TOWARDS WIDER ADOPTION OF XRAY INSTRUMENTATION

As of now, XRay has not found wider adoption as an instrumentation back-end

We are trying to change this by:

1) Enhancing core capabilities
    - Shared library instrumentation
    - Improved control over instrumentation points

2) Demonstrating benefits by integrating XRay into established HPC tools
    - Score-P instrumentation back-end based on XRay
    - XRay support for Extrae[1] & TALP within CaPI
        - Research tool for compiler-assisted selective instrumentation

https://github.com/tudasc/CaPI

[1] https://tools.bsc.es/extrae

TOWARDS WIDER ADOPTION OF XRAY INSTRUMENTATION

As of now, XRay has not found wider adoption as an instrumentation back-end

We are trying to change this by:

1) Enhancing core capabilities
   - **Shared library instrumentation**
   - Improved control over instrumentation points

2) Demonstrating benefits by integrating XRay into established HPC tools
   - **Score-P instrumentation back-end based on XRay**
   - XRay support for Extrae[1] & TALP within CaPI
     - Research tool for compiler-assisted selective instrumentation

CaPI

https://github.com/tudasc/CaPI

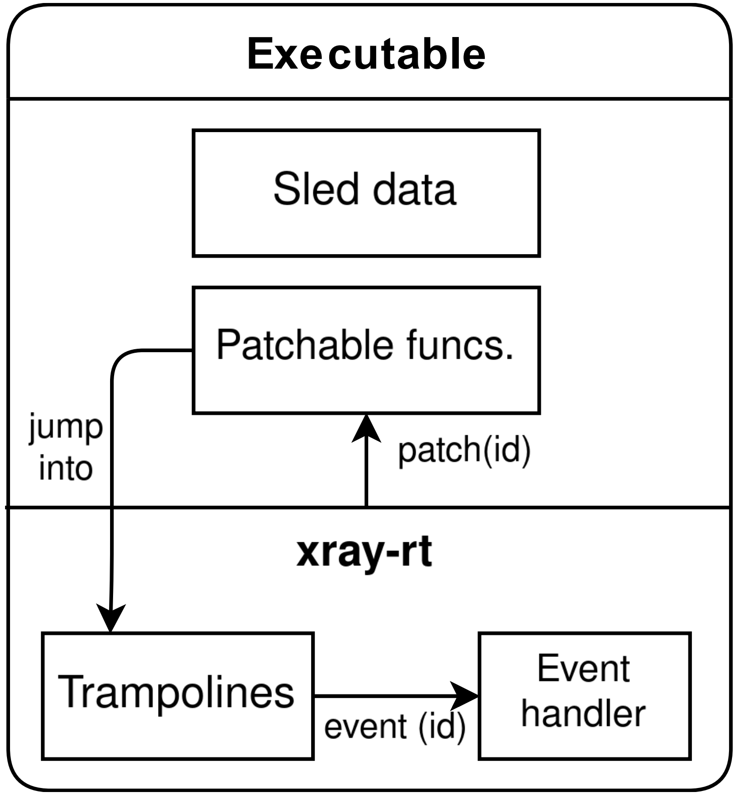
[1] https://tools.bsc.es/extrae

ADDING SHARED LIBRARY SUPPORT

- We recently upstreamed support for DSO instrumentation
  - Available in LLVM 20
  - Enabled with `-fxray-shared` flag
  - See pull request for details: https://github.com/llvm/llvm-project/pull/113548

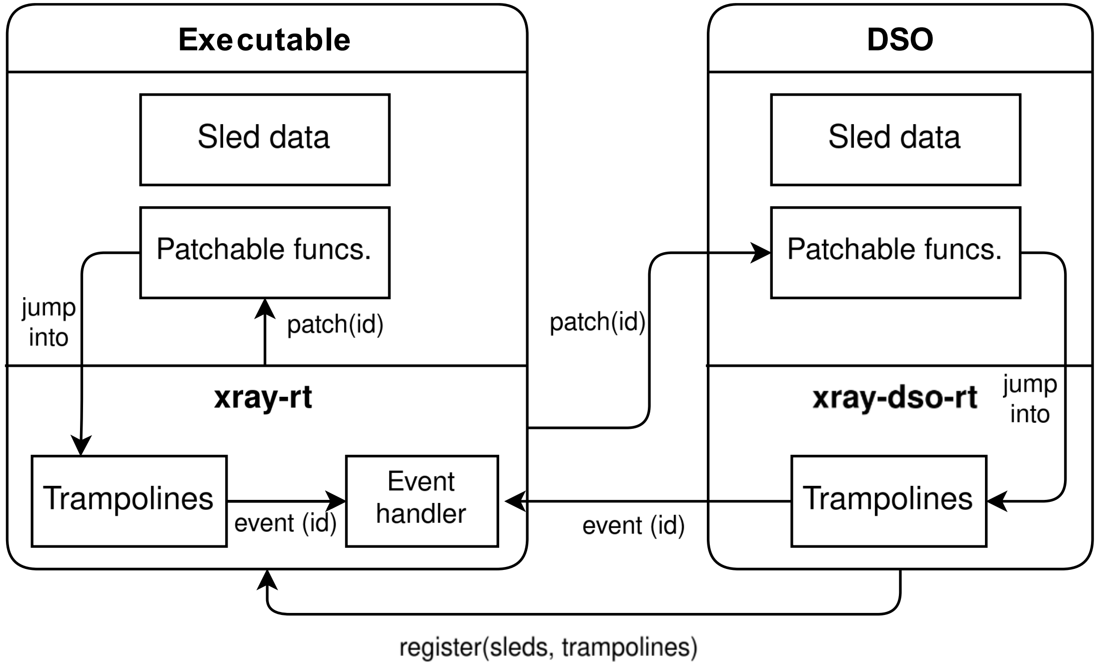
- Support in built-in XRay tracing modes is work-in-progress
  - Currently cannot properly resolve function IDs from DSOs
  - RFC: https://discourse.llvm.org/t/rfc-xray-adding-runtime-symbol-resolution-to-xray/85397
  - PR with prototype implementation: https://github.com/llvm/llvm-project/pull/133269

PATCHING (EXECUTABLE ONLY)

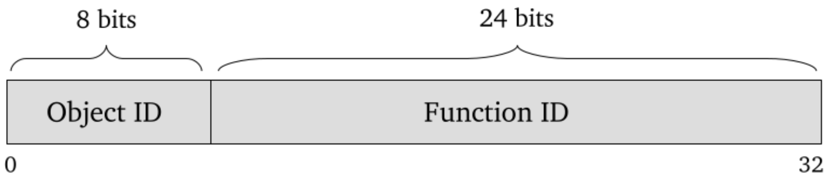


- `Sled data` contains information about patchable functions
- Functions identified by unique 32-bit function ID
- Function ID passed to the event handler on invocation

PATCHING SHARED LIBRARIES



- On load, each instrumented DSO self-registers with the main runtime
- Receives dynamic object ID
- Patching orchestrated by main runtime
- Patched functions invoke event handler with packed ID

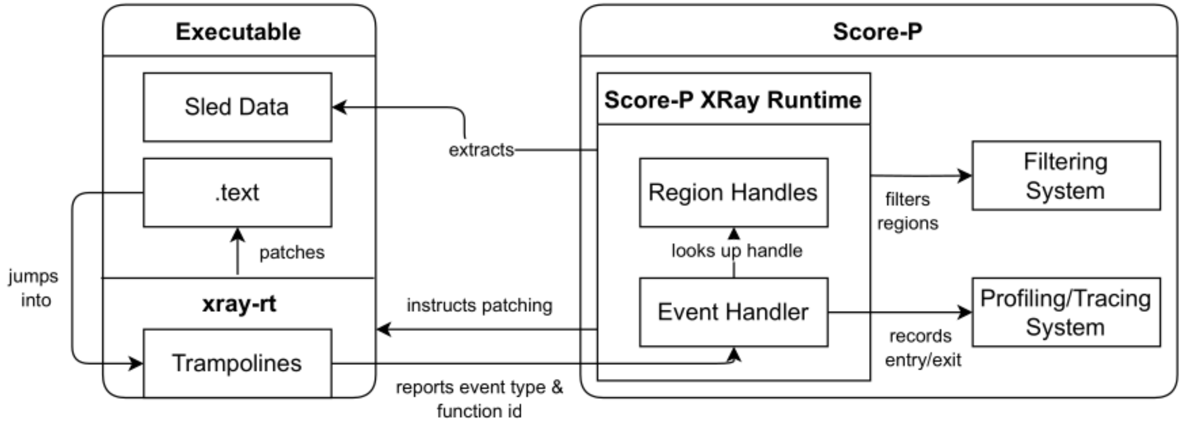


Packed function ID layout

- Other changes: extended XRay API, relocatable trampoline code
- Support implemented for X86_64 and Aarch64
  → See https://github.com/llvm/llvm-project/pull/115300 for reference to add support for other targets

SCORE-P WITH XRAY BACK-END

- Available here: https://github.com/tudasc/scorep-xray
- Basic usage:
  - Configure with `--enable-xray-plugin`
  - Compile with `scorep-clang++ wrapper`
- DSO instrumentation is work-in-progress

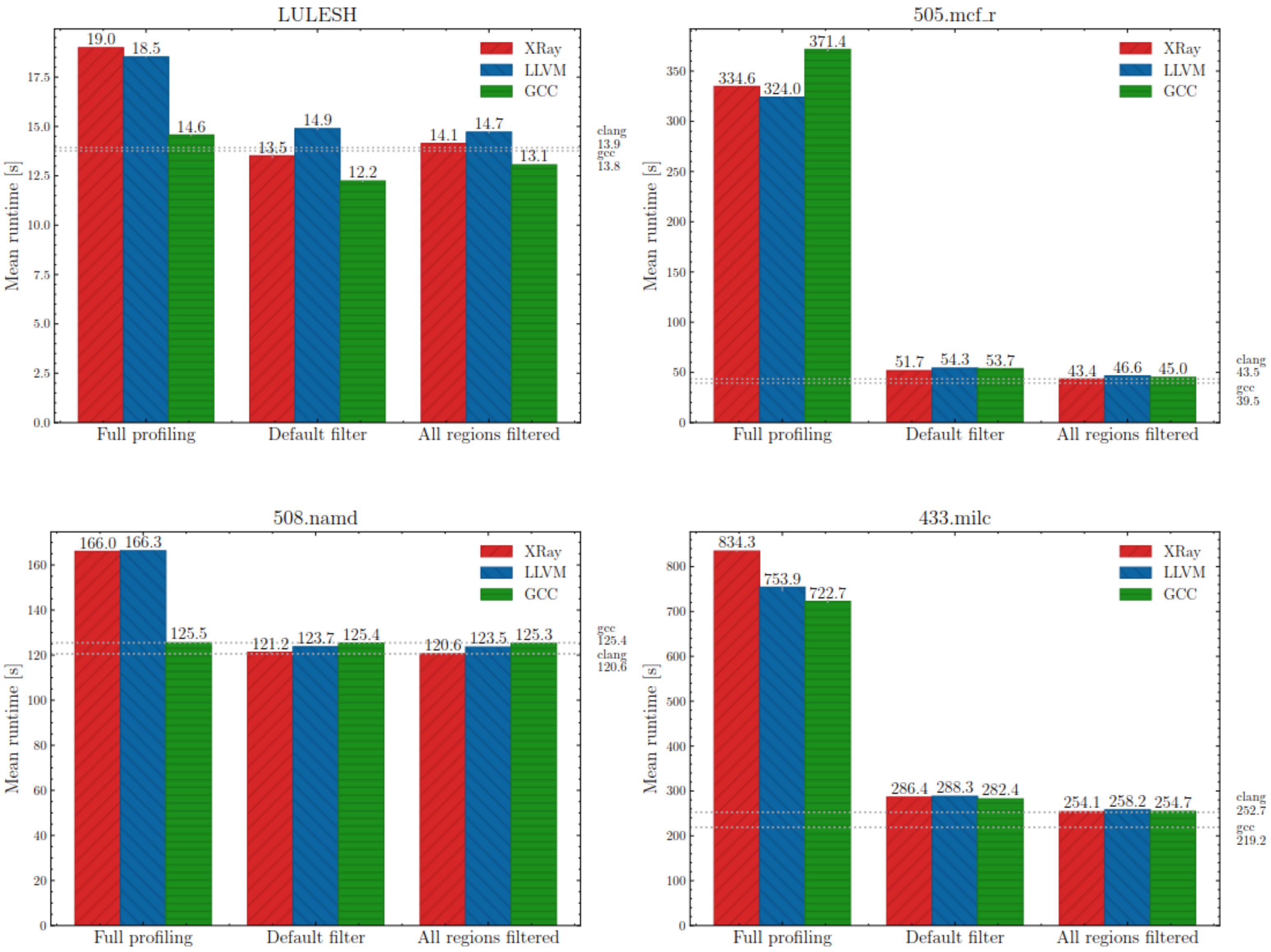


Kreutzer, S., Adelmann, P., Bischof, C. (2025) "**A Runtime-Adaptable Instrumentation Back-End for Score-P**", Proceedings of the 2024 International Parallel Tools Workshop (accepted and to appear)

SCORE-P EVALUATION

- Question: how does XRay’s performance compare to a polished, tool-specific instrumentation plugin?

- Comparing XRay to Score-P’s instrumentation plugins for Clang & GCC
  - Tool-specific instrumentation API
  - Embedded region information
  - Fast dynamic filtering at call site


LULESH

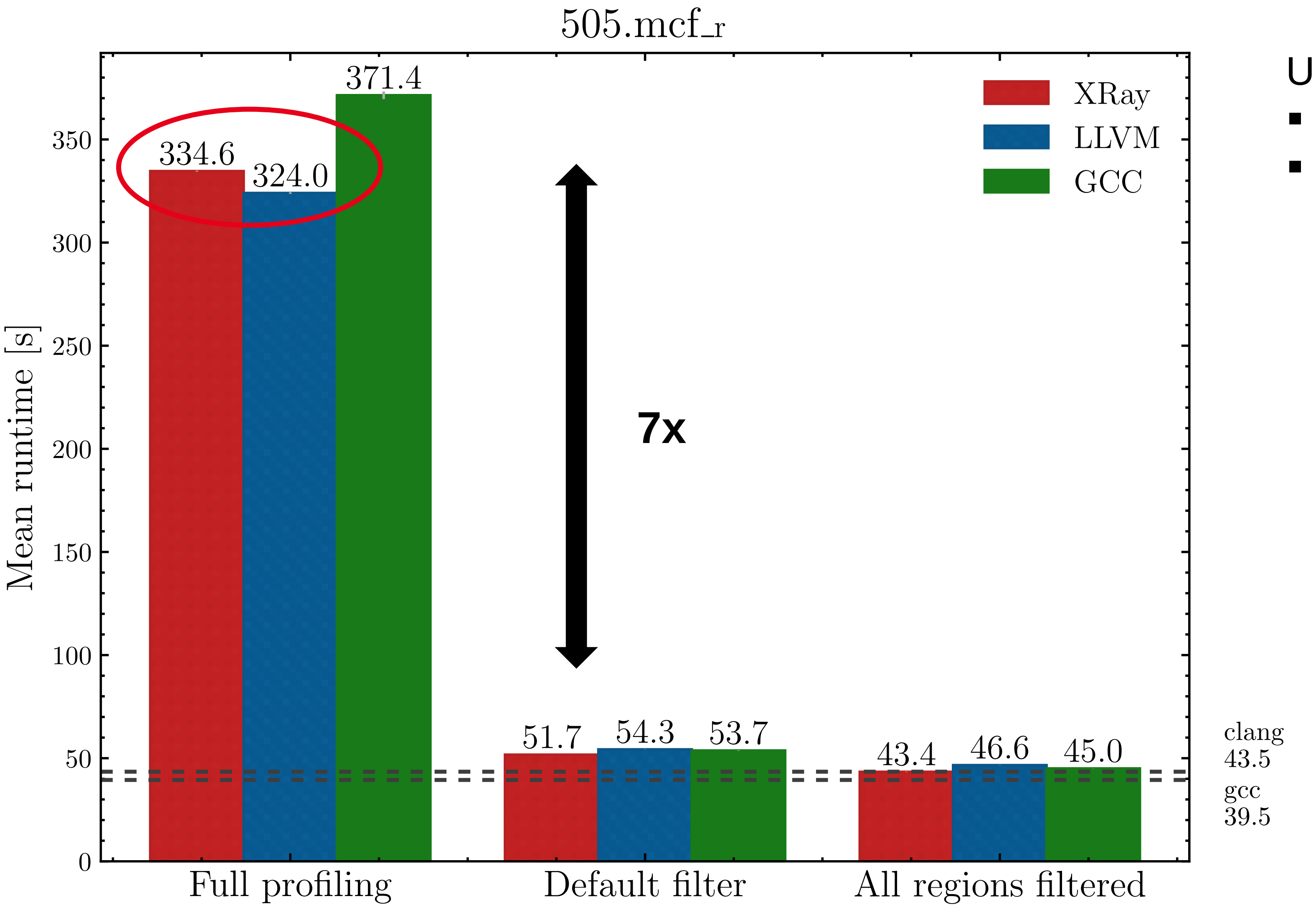
505.mcf_r

508.namd

433.milc

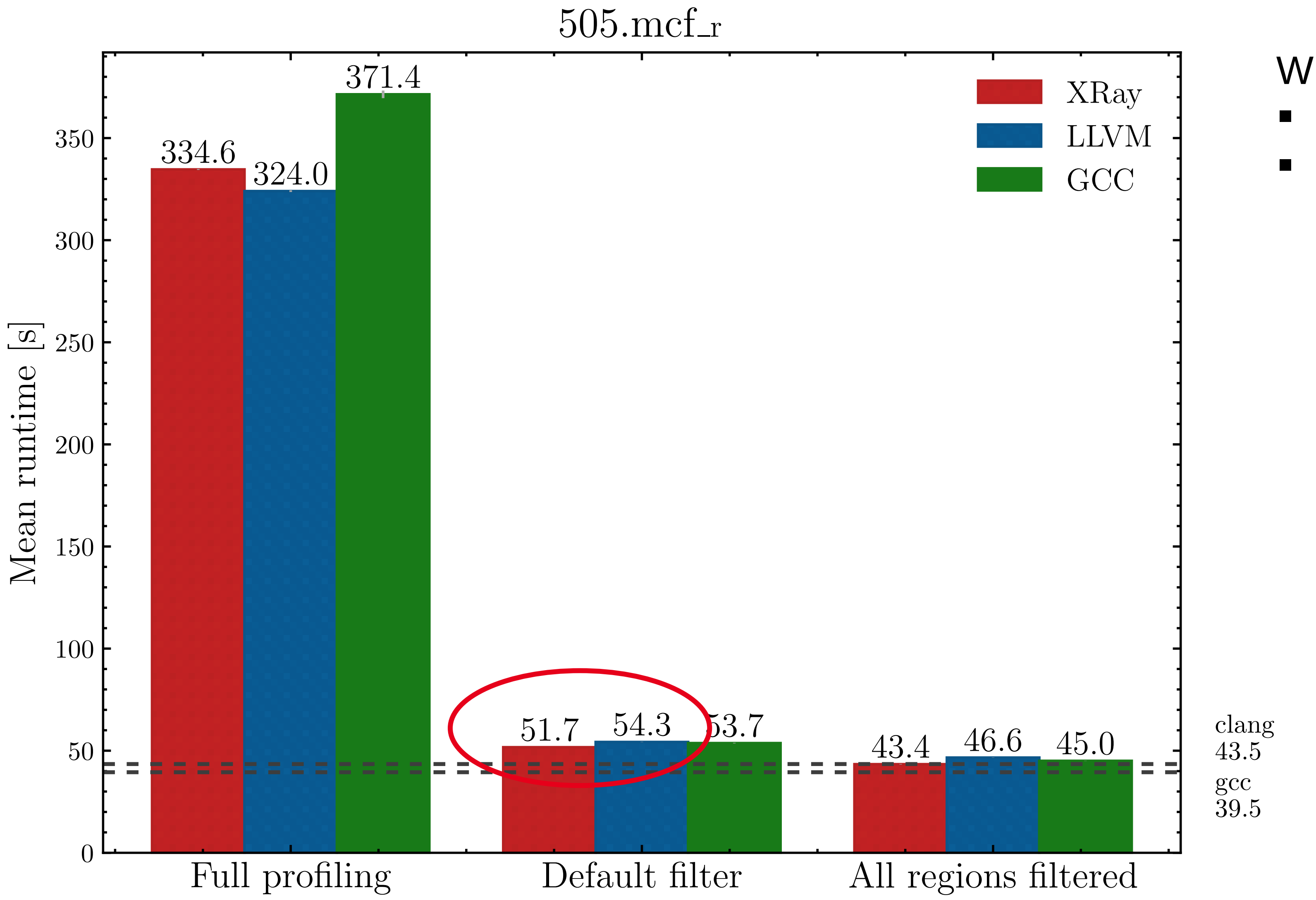
Overhead results (serial execution) from LULESH and selected SPEC benchmarks[1].

[1] Benchmarks from SPEC CPU®2006 and SPEC CPU®2017. Results measure the overhead of the instrumentation method only and do not constitute compliant results according to the SPEC fair use rules. Specifically, we do not make any claims regarding the performance of the underlying benchmarks.



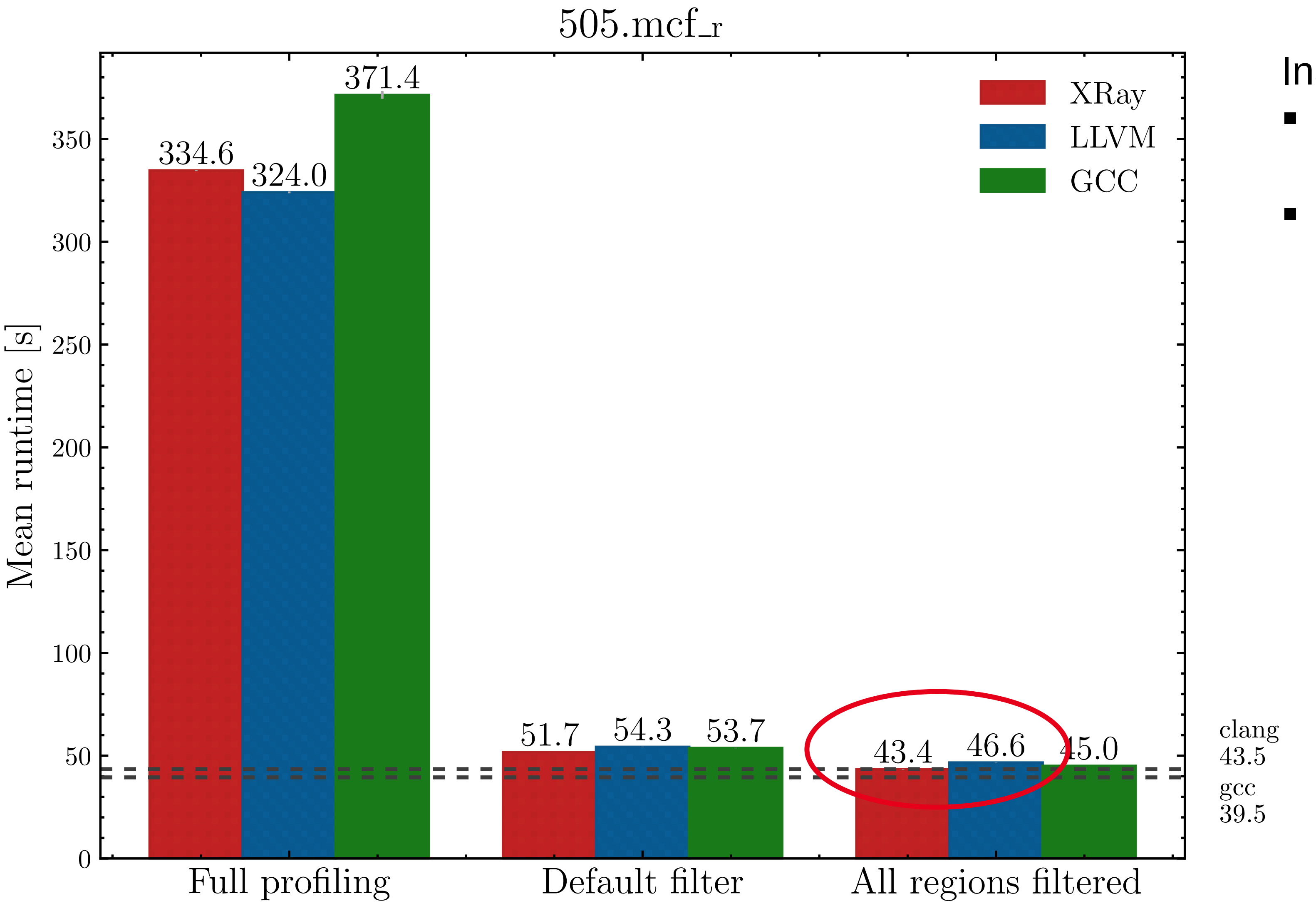
Unfiltered configuration:

- Slightly more overhead with XRay
- Only significant in configurations with too much overhead to be useful



With dynamic filtering:

- Filter generated using `scorep-score`
- XRay matches or outperforms the static instrumentation



Inactive instrumentation:

- XRay does not incur measurable overhead
- Compared to up to 6% with plugin

INCREASING INSTRUMENTATION FLEXIBILITY

- XRay instrumentation points: post-inlining and function-level only
- Can be manually inserted using custom events (`__xray_customevent` function)
  - Requires custom handler
- Possible improvement: explicit support for loop and pre-inline instrumentation
  - Via extended instrumentation flags (e.g., `-fxray-instrument-loops`)
  - Better support for declaring custom regions on the IR level

CONCLUSION

- There are many ways to perform automatic function instrumentation, with different drawbacks and advantages
- XRay combines a lot of the advantages and is easy to integrate
- To make XRay viable for HPC tools, we work on extending core features, e.g. DSO instrumentation
- We demonstrated the advantages of XRay by implementing an instrumentation back-end for Score-P

- Are you using XRay with the built-in tracing libraries or in a custom tool?
I would love to hear about your experience!