

# Score-P

## Introduction

The Score-P measurement infrastructure is a highly scalable and easy-to-use tool suite for profiling, event tracing, and online analysis of HPC applications.

Score-P can be used as an instrumentation tool for Scalasca.

## Installed versions

There are currently two versions of Score-P version 1.2.6 modules installed on Anselm :

- class="s1">scorep/1.2.3-gcc-openmpi>, for usage with GNU Compiler> and OpenMPI{.internal> ,
- class="s1">scorep/1.2.3-icc-impi>, for usage with Intel Compiler> and Intel MPI>.

## Instrumentation

There are three ways to instrument your parallel applications in order to enable performance data collection :

1. Automated instrumentation using compiler
2. Manual instrumentation using API calls
3. Manual instrumentation using directives

### Automated instrumentation

is the easiest method. Score-P will automatically add instrumentation to every routine entry and exit using compiler hooks, and will intercept MPI calls and OpenMP regions. This method might, however, produce a large number of data. If you want to focus on profiler a specific regions of your code, consider using the manual instrumentation methods. To use automated instrumentation, simply prepend scorep to your compilation command. For example, replace :

```
$ mpif90 -c foo.f90
$ mpif90 -c bar.f90
$ mpif90 -o myapp foo.o bar.o
```

with :

```
$ scorep mpif90 -c foo.f90
$ scorep mpif90 -c bar.f90
$ scorep mpif90 -o myapp foo.o bar.o
```

Usually your program is compiled using a Makefile or similar script, so it is advisable to add the scorep command to your definition of variables `CC`, `CXX`, `class="monospace">FCC` etc.

It is important that scorep is prepended also to the linking command, in order to link with Score-P instrumentation libraries.

### >Manual instrumentation using API calls

To use this kind of instrumentation, use scorep with switch `-user`. You will then mark regions to be instrumented by inserting API calls.

An example in C/C++ :

```
#include <scorep/SCOREP_User.h>
void foo()
{
    SCOREP_USER_REGION_DEFINE( my_region_handle )
    // more declarations
    SCOREP_USER_REGION_BEGIN( my_region_handle, "foo", SCOREP_USER_REGION_TYPE_COMMON )
    // do something
    SCOREP_USER_REGION_END( my_region_handle )
}
```

and Fortran :

```
#include "scorep/SCOREP_User.inc"
subroutine foo
    SCOREP_USER_REGION_DEFINE( my_region_handle )
    ! more declarations
    SCOREP_USER_REGION_BEGIN( my_region_handle, "foo", SCOREP_USER_REGION_TYPE_COMMON )
    ! do something
    SCOREP_USER_REGION_END( my_region_handle )
end subroutine foo
```

Please refer to the documentation for description of the API.

### >Manual instrumentation using directives

This method uses POMP2 directives to mark regions to be instrumented. To use this method, use command `scorep -pomp`.

Example directives in C/C++ :

```

void foo(...)
{
    /* declarations */
    #pragma pomp inst begin(foo)
    ...
    if (<condition>)
    {
        #pragma pomp inst altend(foo)
        return;
    }
    ...
    #pragma pomp inst end(foo)
}

```

and in Fortran :

```

subroutine foo(...)
!declarations
!POMP$ INST BEGIN(foo)
...
if (<condition>) then
!POMP$ INST ALTEND(foo)
return
end if
...
!POMP$ INST END(foo)
end subroutine foo

```

The directives are ignored if the program is compiled without Score-P. Again, please refer to the documentation for a more elaborate description.