CDAC HPC Profiler

Abhishek Patil

Outline

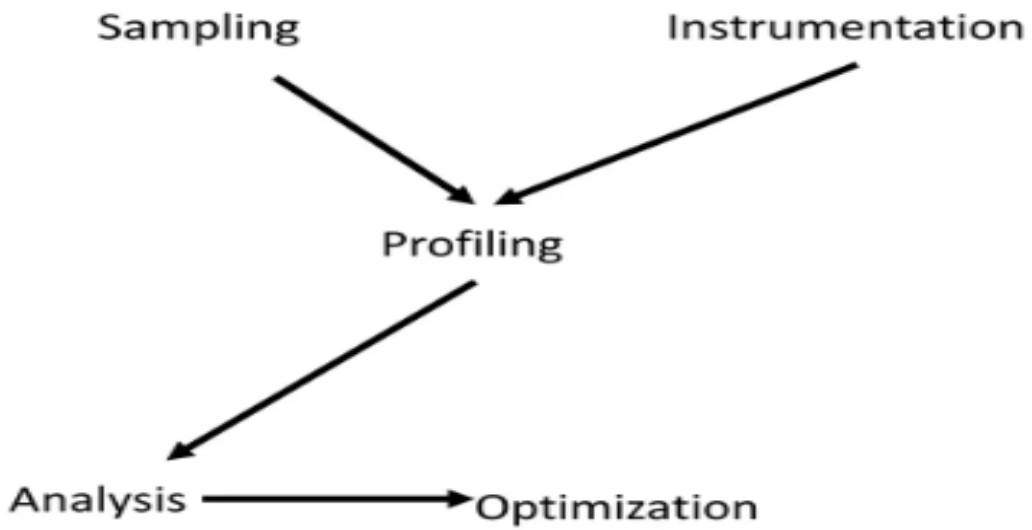
- Introduction
- What is Profiling
- Key Features
- Application Run/Demo
- Screenshots
- Outcome of the experiments

Introduction

- What is Profiling
  - Profiling is the measurement of which parts of your application are consuming a particular computational resource of interest. This could be which methods are using the most CPU time, which lines allocate the most objects, where your CPU cache misses are coming from, etc.
- Reflection of summary information during execution – time consumed, function calls ,
- Reflects performance behaviour of program entities

  functions , loops, basic blocks
- Very good for low cost performance assessment
- Helps to understand performance bottlenecks and hotspots
- Implemented through either
  - Sampling
  - Measurement

Big Picture of Profiling



Brief info about Sampling

- The most common type of profiler is the ***sampling* profiler**.

They work **by interrupting the application** under test periodically in proportion to **the consumption of the resource** we're interested in.

While the program is interrupted the **profiler grabs a snapsho**t of its current state, which includes where in the code it is.

After the **state is captured** the program continues. For the method timing example earlier, a sampling profiler would interrupt the program after a certain amount of time had elapsed and capture its state. It would then aggregate those samples over time to produce a **statistical picture of the state of the application**. You could use the ***percent of samples*** that contained a method of interest to calculate *how much time was spent in that method* (though not the duration of that method).

Brief info about Instrumenting

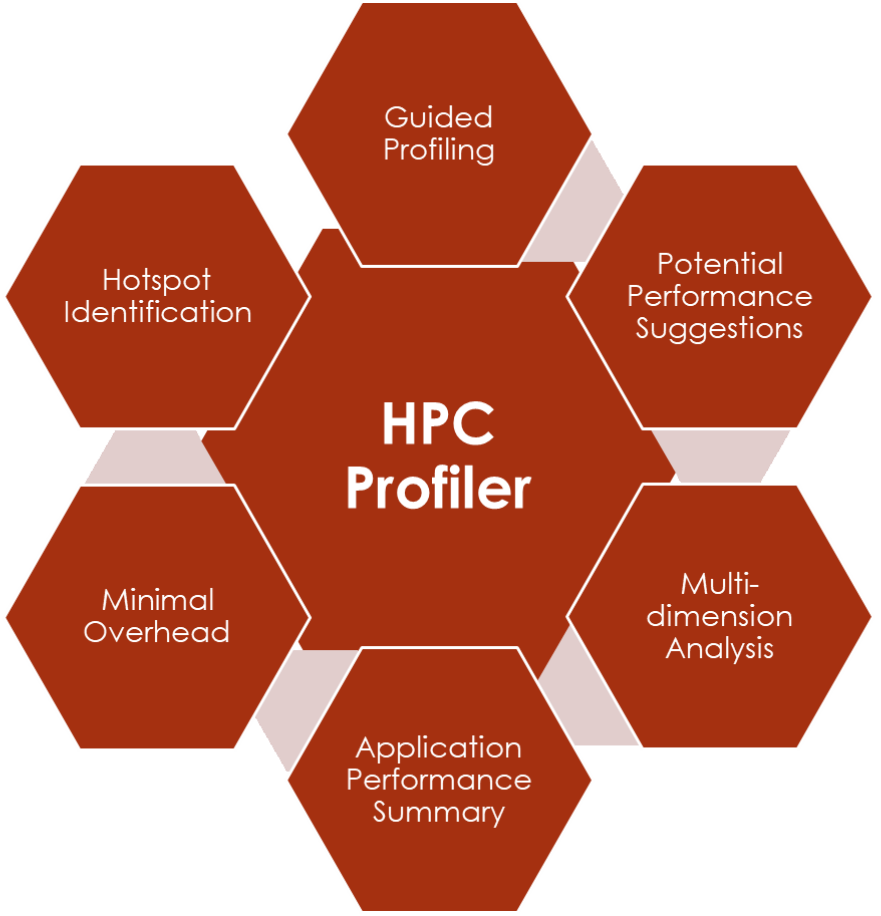
- The first and earliest type are instrumenting profilers.
- They work by *instrumenting* the program under test in order to collect information about the **resource of interest**.

- For example if you wanted to **calculate how much time** methods were taking to execute an instrumenting profiler would **add instructions** to the **beginning and end** of each method to capture the current time which could then be used to reconstruct the **duration spent inside each method**.

What can a profiler tell us

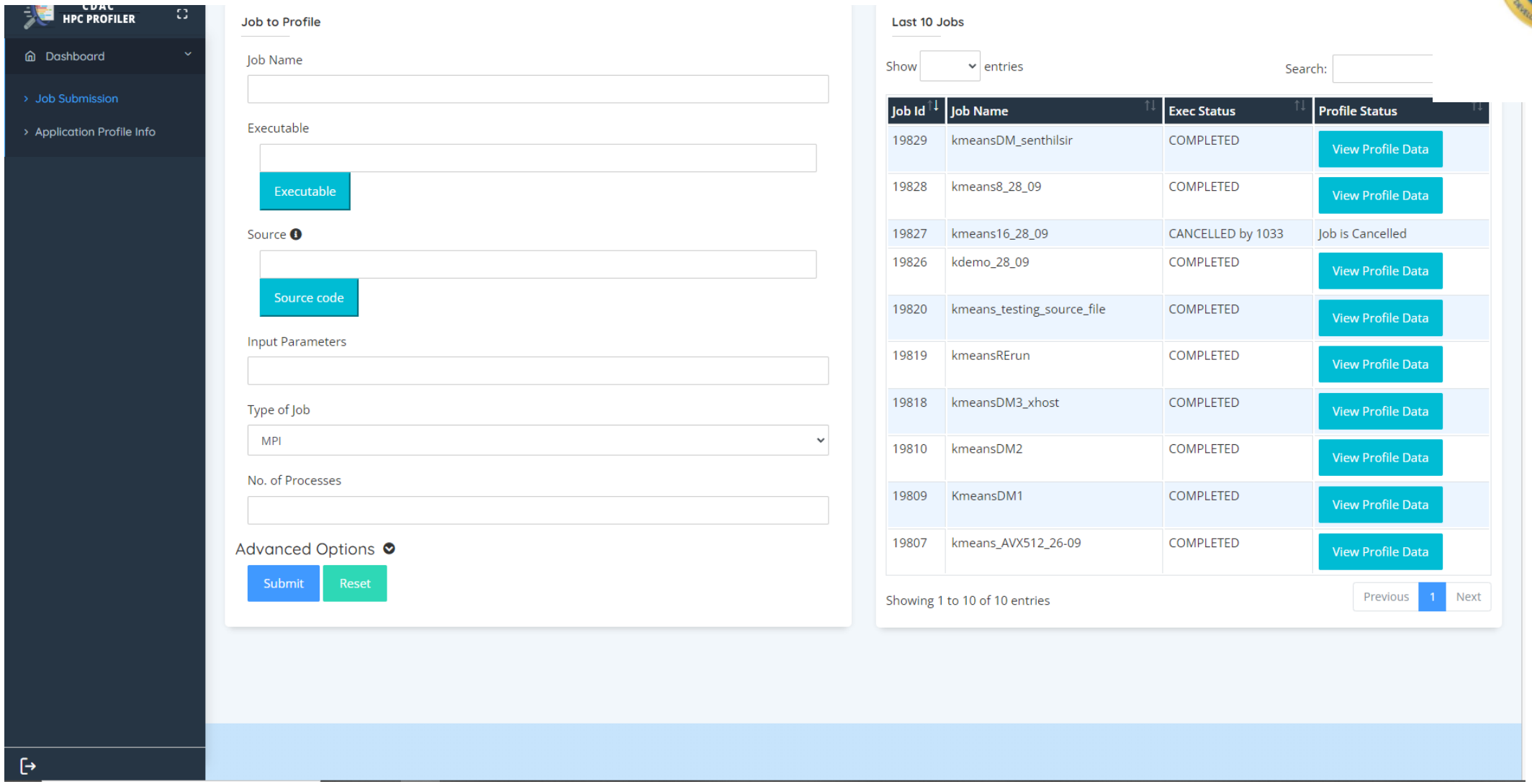
- Processes wise analysis – threads, user functions, loops and blocks
- Memory usage
- Timing information
- Parallel calls (and other library calls)
- Other detailed information like function profiles , vectorization analyses, etc.

HPC Profiler – Key features

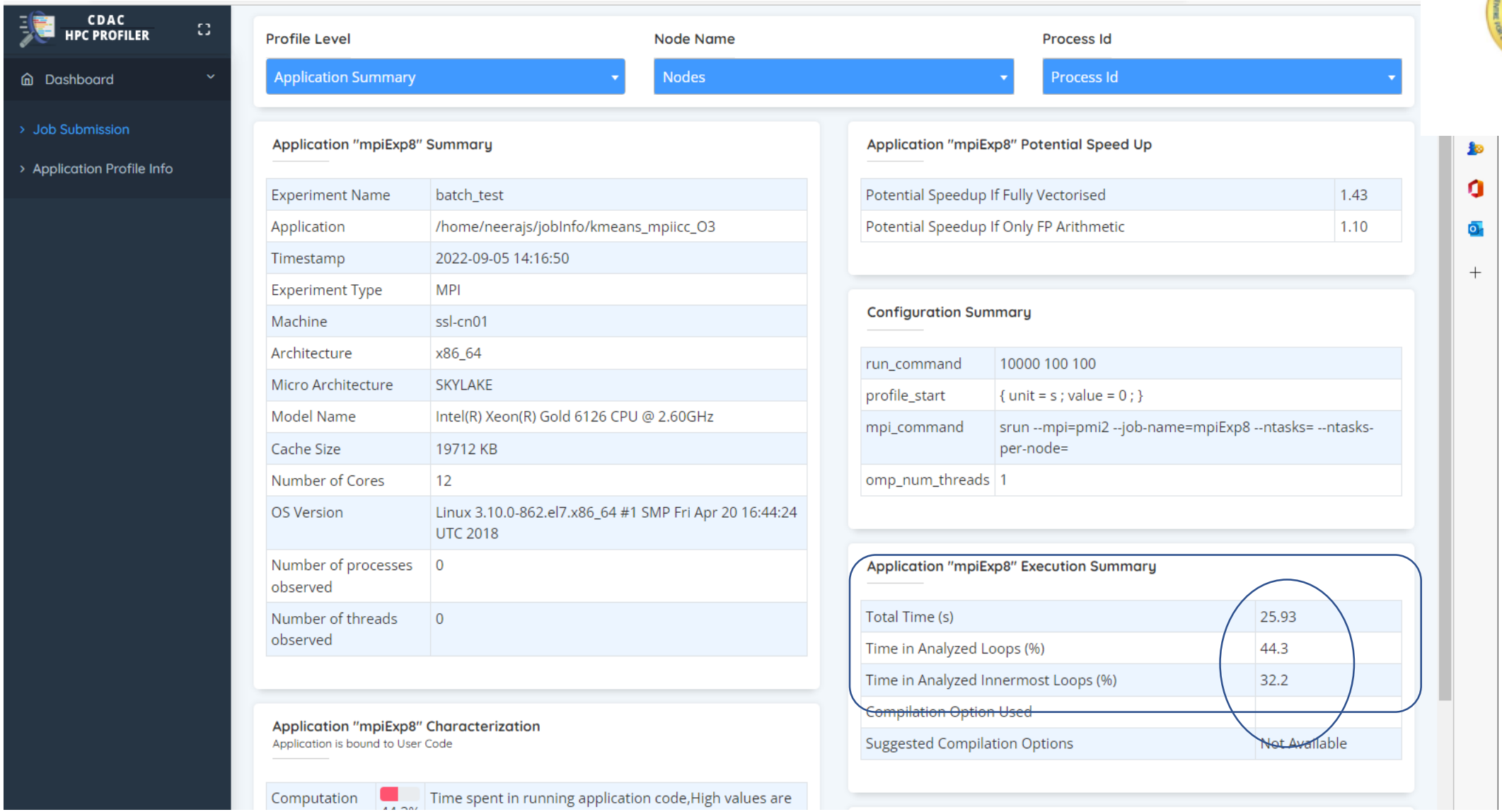


Screenshots

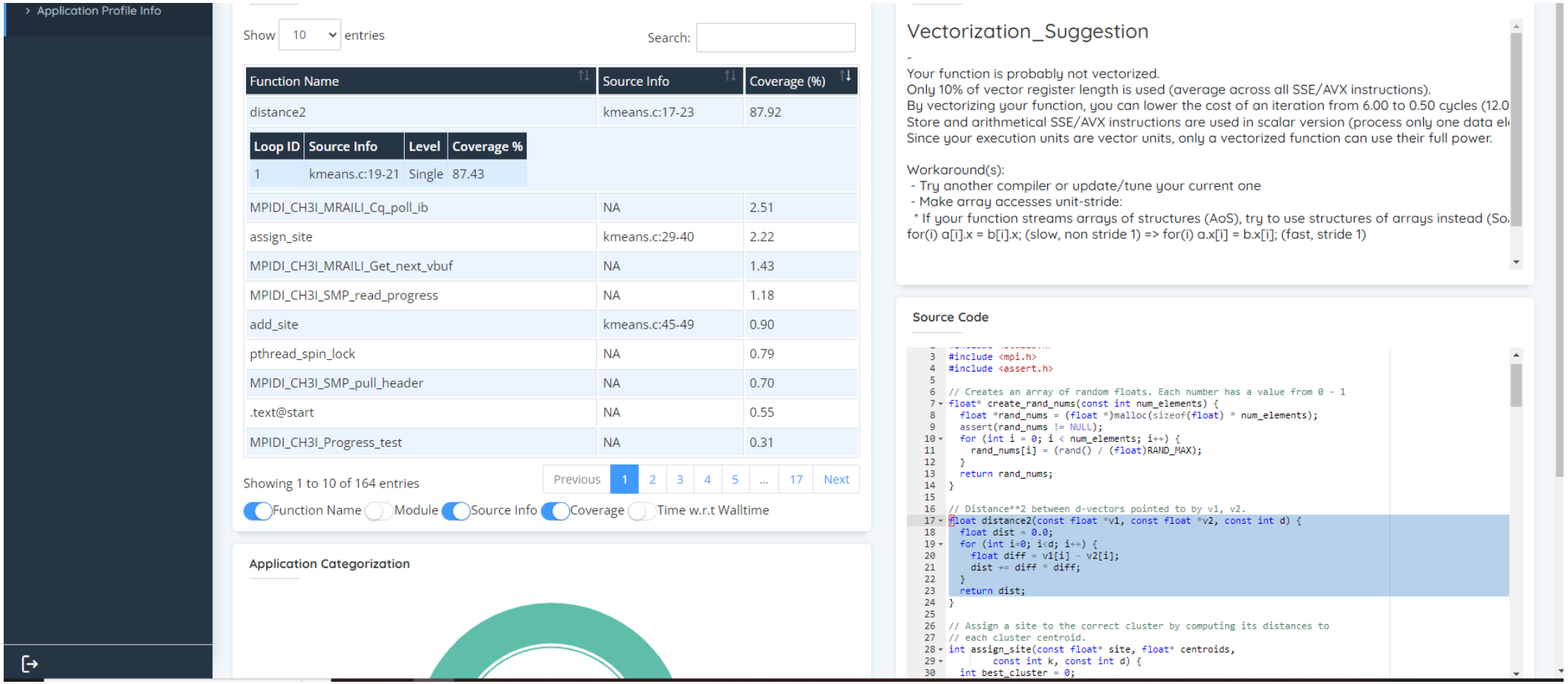
Index page :-


HPC PROFILER
Dashboard
Job Submission
Application Profile Info
Job to Profile
Job Name
Executable
Executable
Source
Source code
Input Parameters
Type of Job
MPI
No. of Processes
Advanced Options
Submit
Reset
Last 10 Jobs
Show entries
Search:
Job Id
Job Name
Exec Status
Profile Status
19829 kmeansDM_senthilsir COMPLETED View Profile Data
19828 kmeans8_28_09 COMPLETED View Profile Data
19827 kmeans16_28_09 CANCELLED by 1033 Job is Cancelled
19826 kdemo_28_09 COMPLETED View Profile Data
19820 kmeans_testing_source_file COMPLETED View Profile Data
19819 kmeansRErun COMPLETED View Profile Data
19818 kmeansDM3_xhost COMPLETED View Profile Data
19810 kmeansDM2 COMPLETED View Profile Data
19809 KmeansDM1 COMPLETED View Profile Data
19807 kmeans_AVX512_26-09 COMPLETED View Profile Data
Showing 1 to 10 of 10 entries
Previous
1
Next

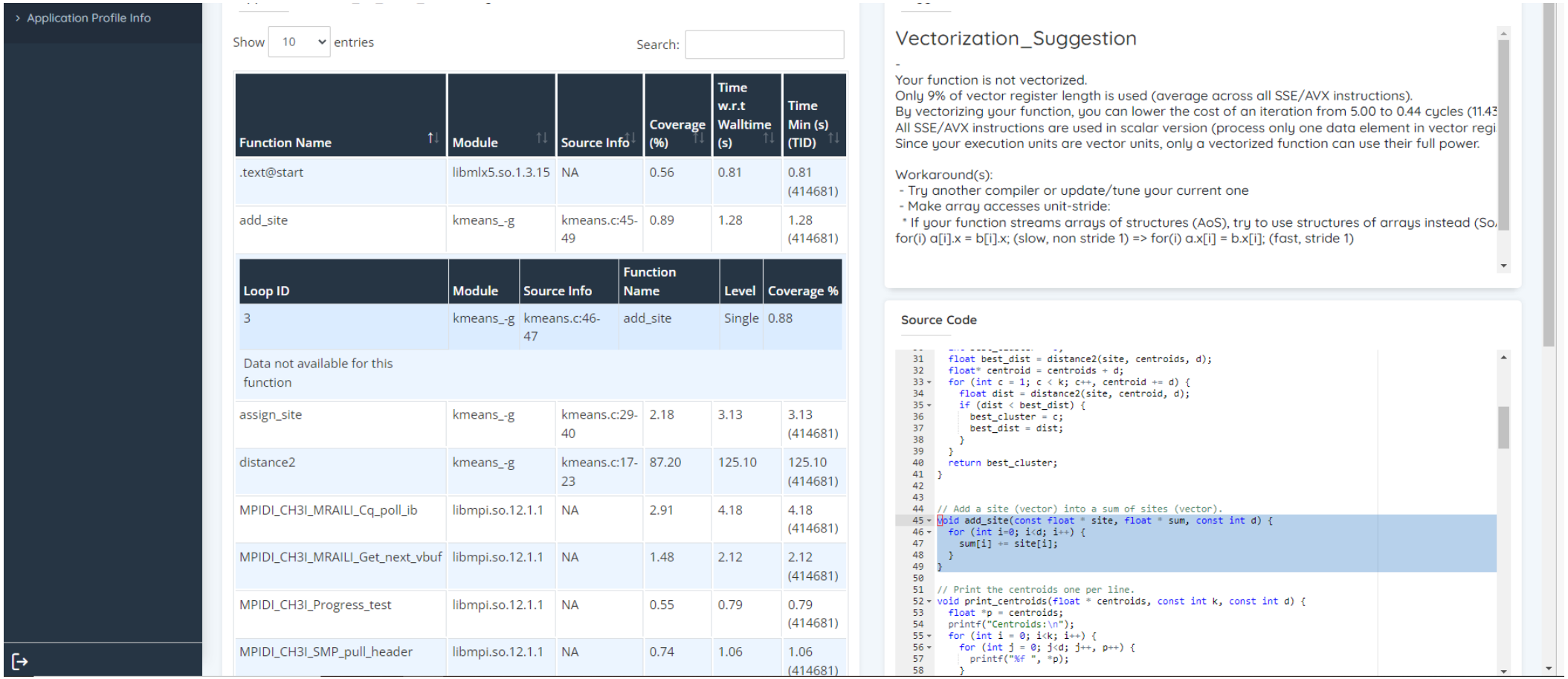
Application summary page :-



Cluster level page :-

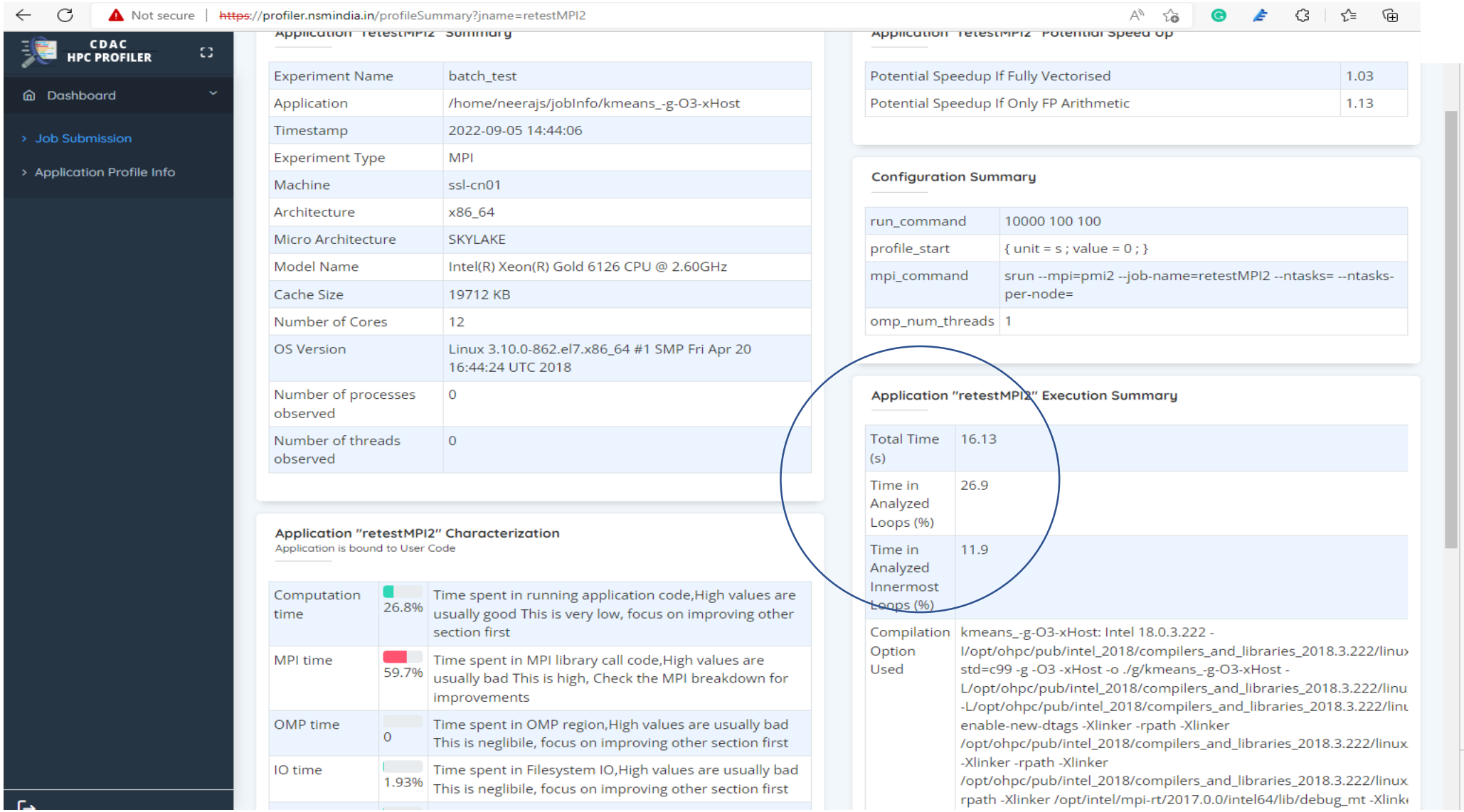


Process level info page :-



The result after implementing suggestions

Outcome of experiment :-



THANK YOU !

For queries contact us at:

nsmss@cdac.in