Performance Analysis Tools

Performance Analysis Tools

- Application performance analysis tools are software solutions designed to monitor, measure, and analyze the performance of applications in order to identify bottlenecks, optimize resource utilization, and improve overall application efficiency. These tools provide insights into various performance metrics such as response time, throughput, latency, error rates, and resource consumption.

Performance Analysis Tools

- Application performance analysis tools are software solutions designed to monitor, measure, and analyze the performance of applications in order to identify bottlenecks, optimize resource utilization, and improve overall application efficiency. These tools provide insights into various performance metrics such as response time, throughput, latency, error rates, and resource consumption.

- When it comes to High-Performance Computing (HPC), application performance analysis tools play a crucial role in optimizing the performance of parallel and distributed applications running on supercomputers and clusters. HPC environments often involve complex computational workflows and large-scale simulations, making it essential to identify and address performance bottlenecks to achieve efficient utilization of resources.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **Performance Profilers:**
      Profilers designed for HPC environments help identify performance hotspots in parallel code. They provide detailed insights into the behavior of individual processes or threads, identifying areas of high computation or communication overhead. Examples of HPC profilers include Intel VTune Amplifier, Scalasca, TAU, and HPCToolkit.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **Tracing Tools**
      Tracing tools capture detailed information about the execution flow and communication patterns within HPC applications. They allow the analysis of performance bottlenecks at a fine-grained level, providing a comprehensive view of resource utilization and inter-process communication. Notable tracing tools in the HPC space include Allinea MAP, Extrae, Score-P, and HPCToolkit.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **Parallel Performance Visualization Tools**
      These tools help visualize and analyze the performance data collected from parallel applications. They offer graphical representations of performance metrics, allowing users to identify patterns, imbalances, and potential scalability issues. Tools like Vampir, Paraver, and VisIt are commonly used for visualizing HPC application performance data.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **MPI Profilers**
      MPI is a popular communication library used in HPC applications to facilitate inter-process communication. MPI profilers specifically focus on analyzing and optimizing MPI communication patterns. Tools such as IPM, mpiP, and TAU provide insights into MPI message traffic, load imbalance, and performance characteristics of message passing.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **I/O Profilers**
      In HPC environments, input/output (I/O) operations can significantly impact application performance. I/O profilers help identify I/O bottlenecks, analyze data access patterns, and optimize I/O performance. Tools like Darshan, PIO, and Scalasca's VampirTrace module offer I/O profiling capabilities tailored for HPC workloads.

Performance Analysis Tools

- Following are some application performance analysis tools specifically relevant to HPC:
    - **Performance Analysis Frameworks**
      Performance analysis frameworks provide a comprehensive suite of tools and methodologies for analyzing HPC application performance. They often combine multiple analysis techniques, including profiling, tracing, and visualization, to offer a holistic approach to performance optimization. Examples include the Performance Application Programming Interface (PAPI), PerfSuite, and the Scalable Performance Analysis of Parallel Codes (Scalasca) framework.

Intel VTune Amplifier

- Intel VTune Amplifier is a powerful performance profiling tool designed to analyze the performance of applications running on Intel architecture-based systems, including CPUs, GPUs, and FPGAs. It provides deep insights into the execution behavior of applications, allowing developers and performance engineers to identify performance bottlenecks, optimize code, and improve overall application efficiency.

Intel VTune Amplifier

- Intel VTune Amplifier is a powerful performance profiling tool designed to analyze the performance of applications running on Intel architecture-based systems, including CPUs, GPUs, and FPGAs. It provides deep insights into the execution behavior of applications, allowing developers and performance engineers to identify performance bottlenecks, optimize code, and improve overall application efficiency.

Following are some key features and capabilities:

a) Performance Profiling
b) Advanced Analysis Techniques
c) Threading Analysis
d) GPU Analysis
e) System-wide Profiling
f) Performance Analysis on Cloud and Remote Systems
g) Integration with Development Environment
h) Cross-Platform Support

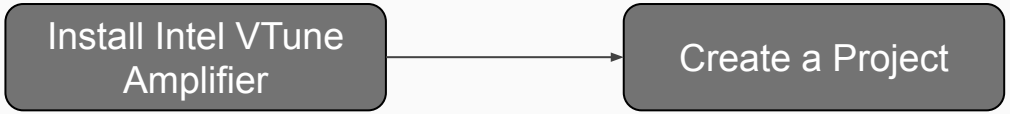
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:

Install Intel VTune Amplifier

Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:

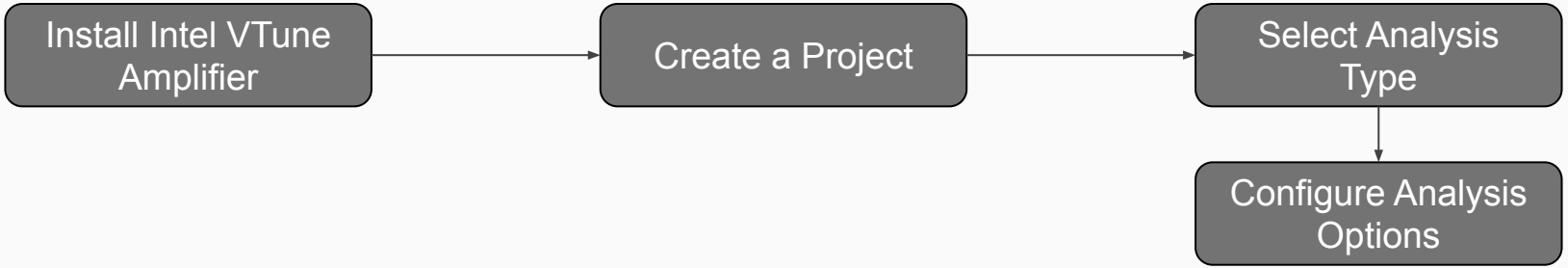


Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:

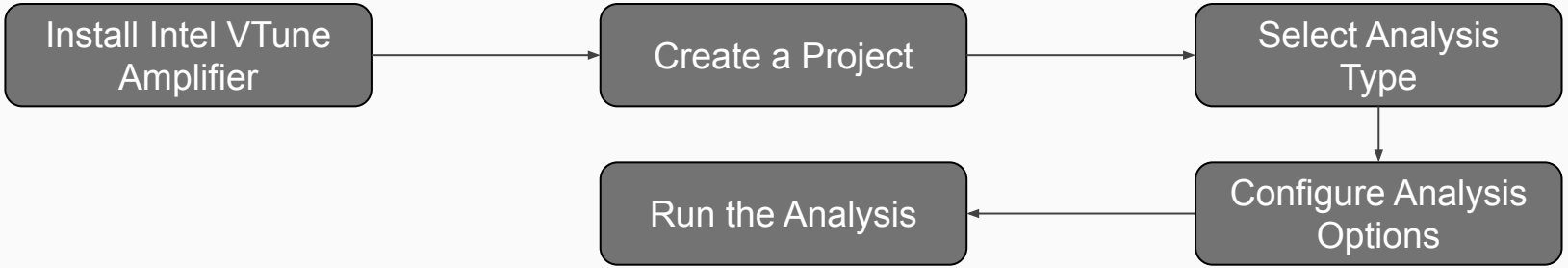
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



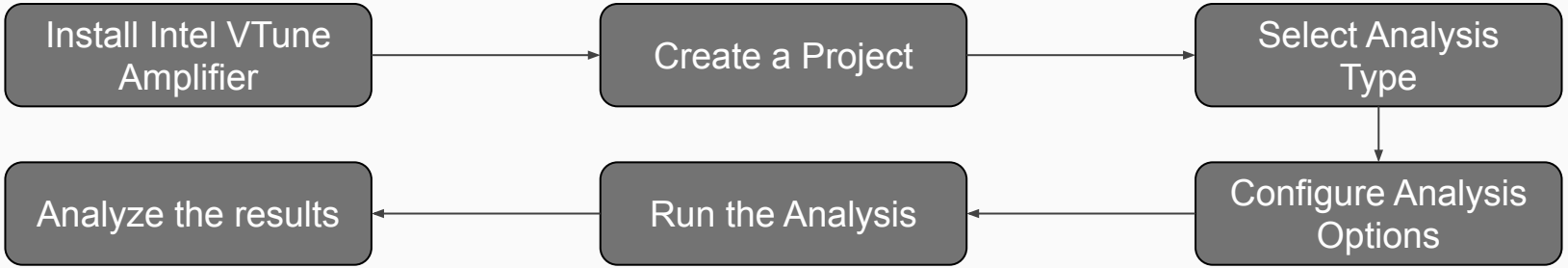
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



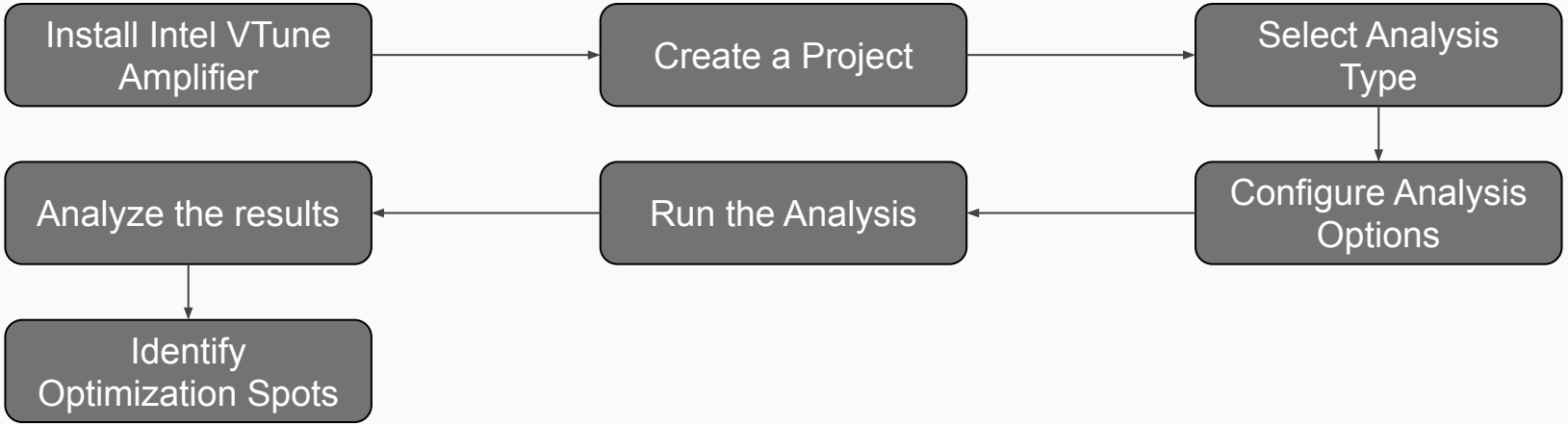
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



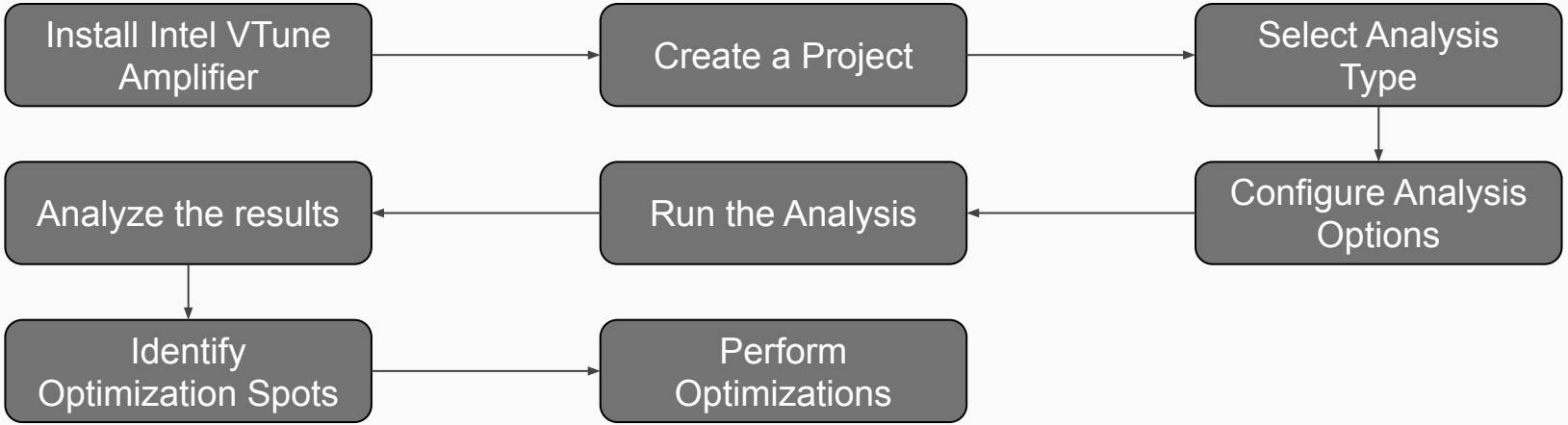
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



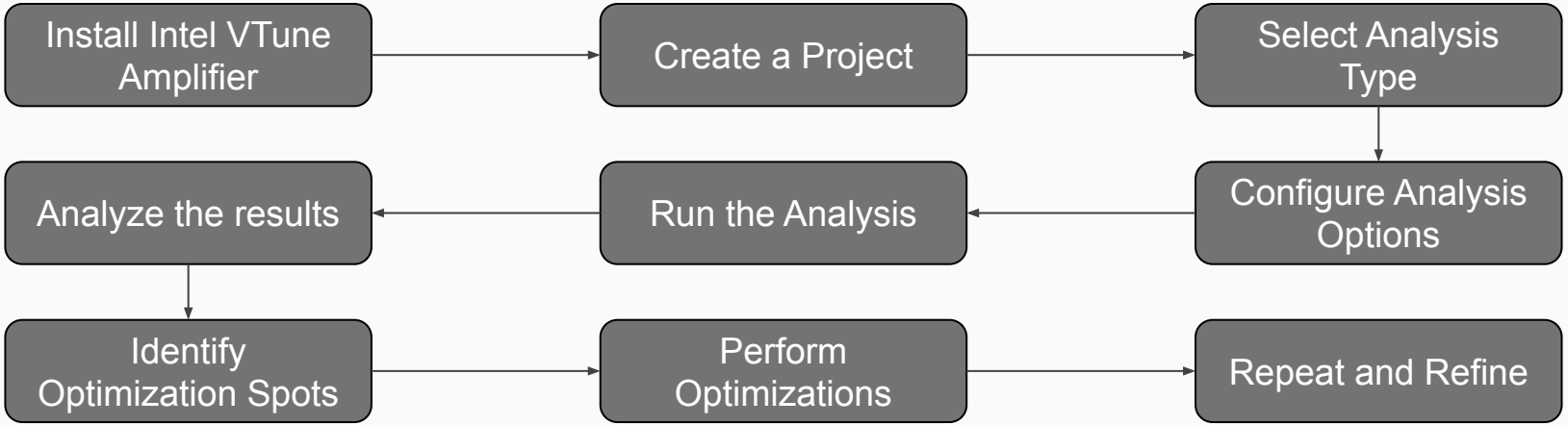
Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



Intel VTune Amplifier

- Following is a general overview of the steps involved in using Intel VTune Amplifier to profile and analyze the performance of any application:



Intel VTune Amplifier

- Intel VTune command syntax:

```
vtune  <-action>  [-action-option]  [-global-option]  [[--]  <target>  [target-options]]
```

-action	: The action to perform, such as collect or report.
-action-option	: Action-options modify behavior specific to the action.
-global-option	: Global-options modify behavior in the same manner for all actions.
target	: The target application to analyse.
target-options	: The options of the application.

More details here:
https://www.intel.com/content/www/us/en/docs/vtune-profiler/user-guide/2023-0/command-syntax.html

Intel VTune Amplifier

- Following are few popular analysis types available for Intel VTune Amplifier:

  - performance-snapshot
  - hotspots
  - memory-consumption
  - memory-access
  - threading
  - hpc-performance
  - io

More details here:
https://www.intel.com/content/dam/develop/external/us/en/documents/vtune-profiler-cheat-sheet.pdf

Workout 1

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any FFTW library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for Intel Vtune Amplifier:

1) Use the application LAMMPS as
the target application for profiling.
2) Execute Intel VTune for the above listed analysis types.
3) Observe the reports generated.

Intel Trace Analyzer and Collector

- Intel ITAC (Intel Trace Analyzer and Collector) is a performance profiling and analysis tool specifically designed for parallel applications. It helps developers and performance engineers analyze the behavior of parallel programs by capturing and analyzing trace data, enabling them to understand performance characteristics, identify bottlenecks, and optimize their applications.

Intel Trace Analyzer and Collector

- Intel ITAC (Intel Trace Analyzer and Collector) is a performance profiling and analysis tool specifically designed for parallel applications. It helps developers and performance engineers analyze the behavior of parallel programs by capturing and analyzing trace data, enabling them to understand performance characteristics, identify bottlenecks, and optimize their applications.

Following are some key features and capabilities:

a) Parallel Trace Collection
b) Tracing Modes
c) Flexible Instrumentation Options
d) Distributed Computing Support
e) Trace Visualization
f) Performance Analysis Capabilities
g) Integration with Intel Parallel Studio
h) Cross-Platform Support

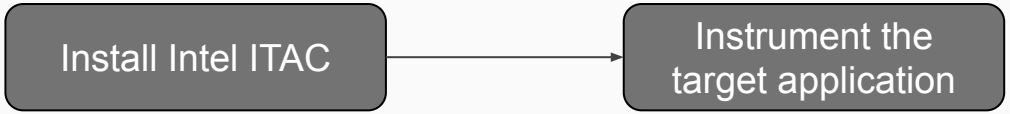
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:

  Install Intel ITAC

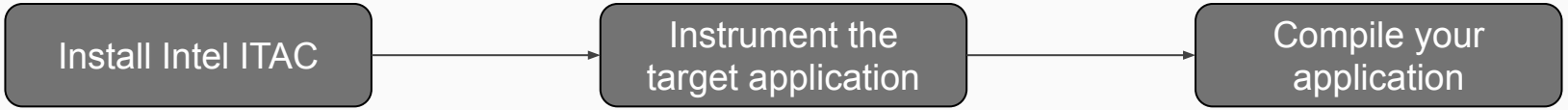
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



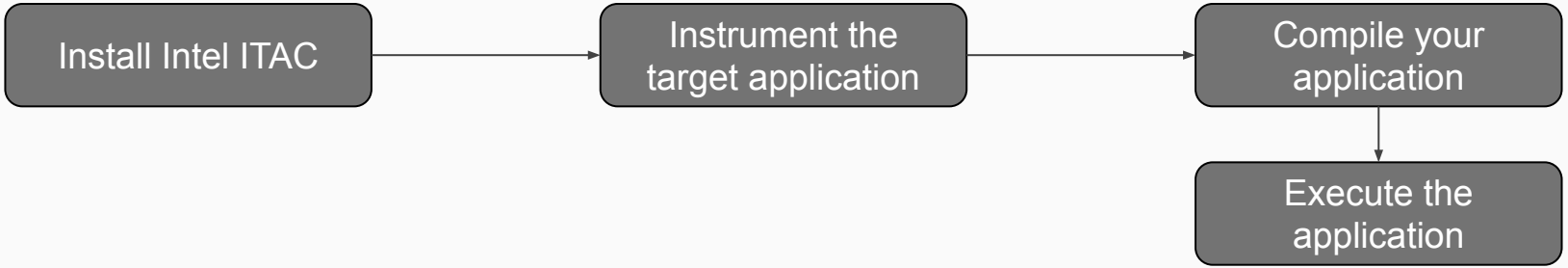
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



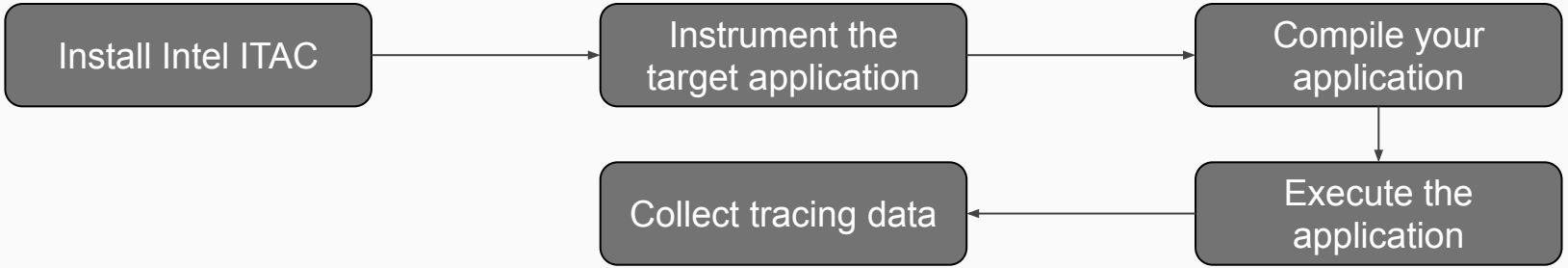
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



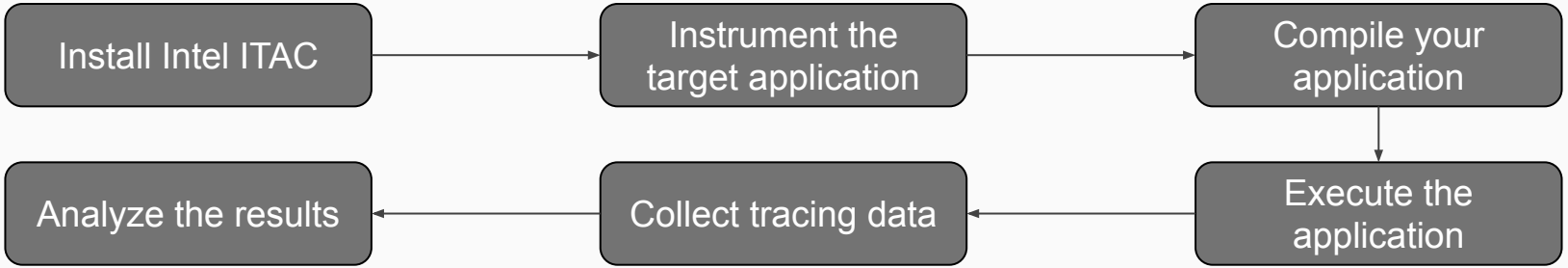
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



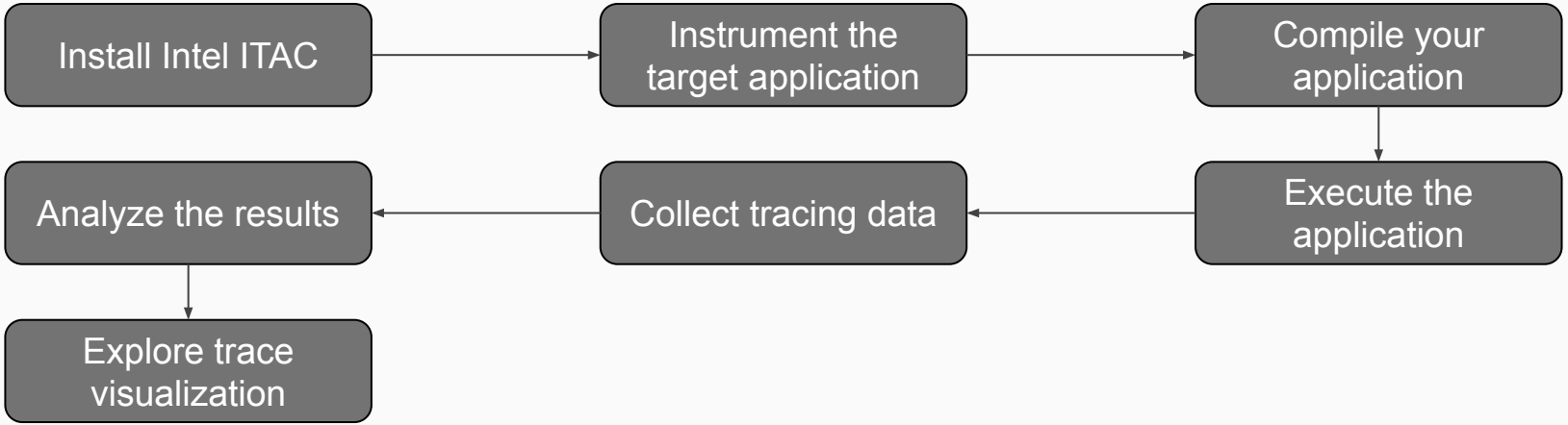
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



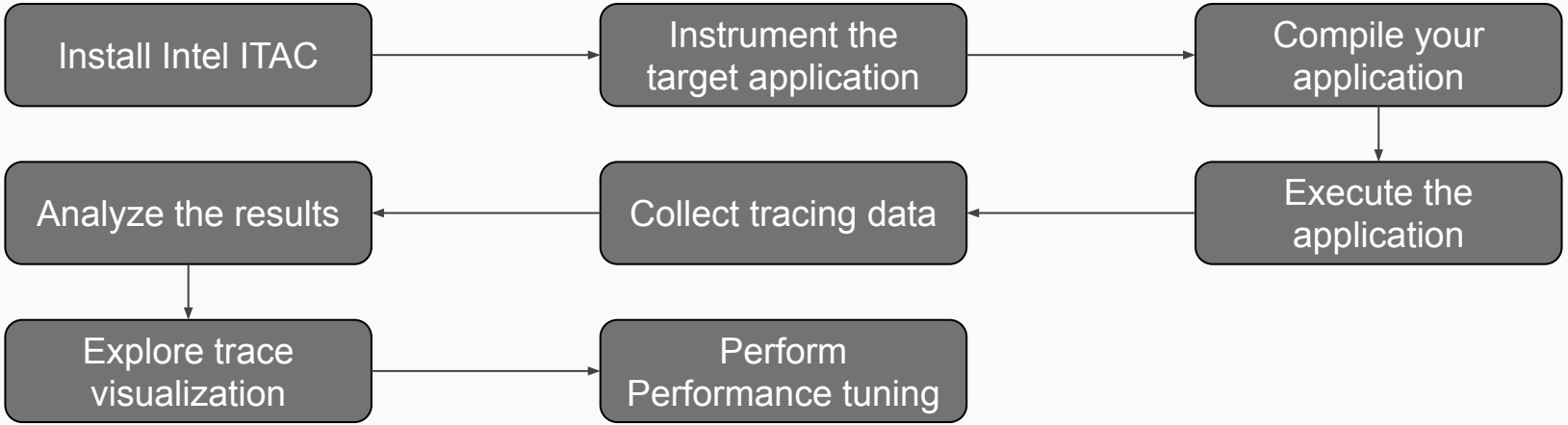
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



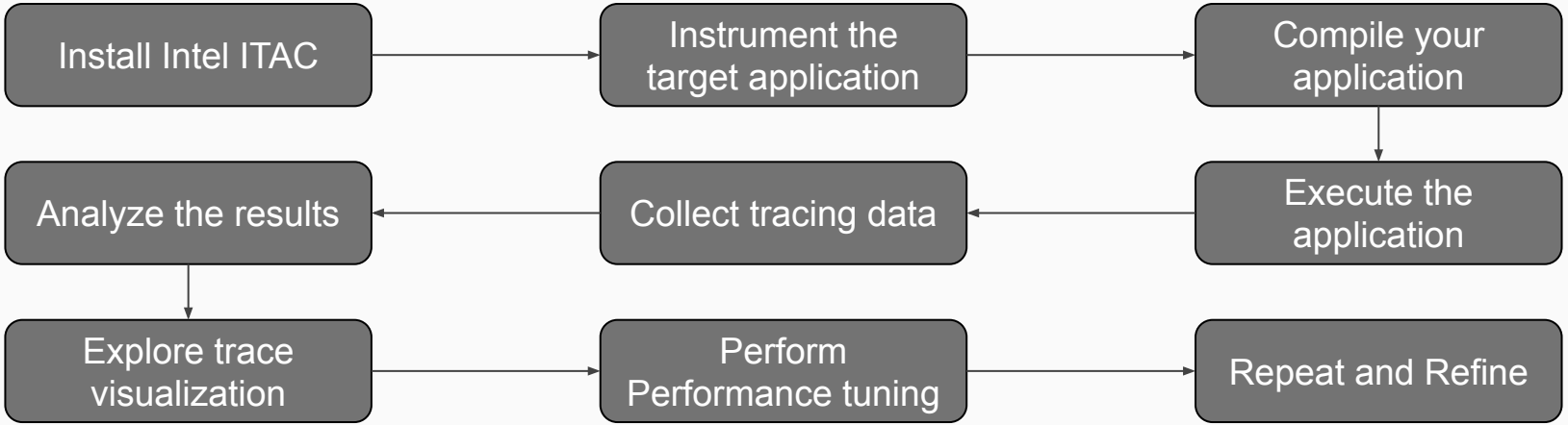
Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



Intel Trace Analyzer and Collector

- To run Intel ITAC (Intel Trace Analyzer and Collector), a performance profiling and analysis tool for parallel applications, follow these general steps:



Intel Trace Analyzer and Collector

- Intel Trace Analyzer and Collector consists of the two tools:
  - a collector with command-line interface (CLI) for tracing and
  - an analyzer with graphical user interface (GUI) for visualizing:

Intel Trace Analyzer and Collector

- Intel Trace Analyzer and Collector consists of the two tools:
  - a collector with command-line interface (CLI) for tracing and
  - an analyzer with graphical user interface (GUI) for visualizing:

- **Intel Trace Collector:**
  Intel Trace Collector is a command-line tool for tracing and analyzing performance of MPI applications. It intercepts all MPI calls and generates .stf trace files that can be analyzed with Intel Trace Analyzer for understanding the application behavior. Intel Trace Collector can also trace non-MPI applications, like socket communication in distributed applications or serial programs.

Intel Trace Analyzer and Collector

- Intel Trace Analyzer and Collector consists of the two tools:
  - a collector with command-line interface (CLI) for tracing and
  - an analyzer with graphical user interface (GUI) for visualizing:

- **Intel Trace Analyzer:**
  Intel Trace Analyzer is a graphical tool that displays and analyzes event trace data, using .stf trace files generated by Intel Trace Collector as input. Intel Trace Analyzer helps you understand the application behavior, detect performance problems and programming errors.

Intel Trace Analyzer and Collector

- Intel ITAC command syntax (Collector):

```
$) source   /path/to/intel/oneapi/setvars.sh
$) mpirun  -trace  <mpi_parameters> <application_executable>
```

Notes:

1) Pre-computed '.stf' files can also be used to analyze the trace.
2) The generated '.stf' files can be shared / moved to different system and analyzed later

Intel Trace Analyzer and Collector

- Intel ITAC command syntax (Analyzer):

```
traceanalyzer  --cli  <action-option>  [tracefile.stf]
```

–cli	: The enable CLI and disable Graphical interface.
action-option	: Action-options modify the trace behavior specific to the action.
tracefile.stf	: Optionally trace files can be supplied to perform analysis

More details here:
https://www.intel.com/content/www/us/en/docs/vtune-profiler/user-guide/2023-0/command-syntax.html

Intel Trace Analyzer and Collector

- Following are few popular analysis types available for Intel Trace Analyzer and Collector:

  - Message Profile
  - Collective Operations Profile
  - Function Profile
  - Request Statistics
  - Summary

More details here:
https://www.intel.com/content/www/us/en/docs/trace-analyzer-collector/user-guide-reference/2023-1/traceanalyzer-command-line-interface.html

Workout 2

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any MPI library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for Intel Trace Analyzer and Collector:

1) Use the application LAMMPS as the target application for profiling.
2) Execute Intel ITAC for the message profile, collective operations profile and function profile.
3) Observe the reports generated.
4) Download and trace the same '.stf' file available over Intel's website and observe the profile.

Intel Advisor

- Intel Advisor is a design and analysis tool for developing performant code.
  Helps you to Design your application for efficient threading, vectorization, and memory use.
  Helps you to identify parts of the code that can be profitably offloaded. Optimize the code for compute and memory.
  Helps you to Create, visualize, analyze computation for heterogeneous algorithms.

Intel Advisor

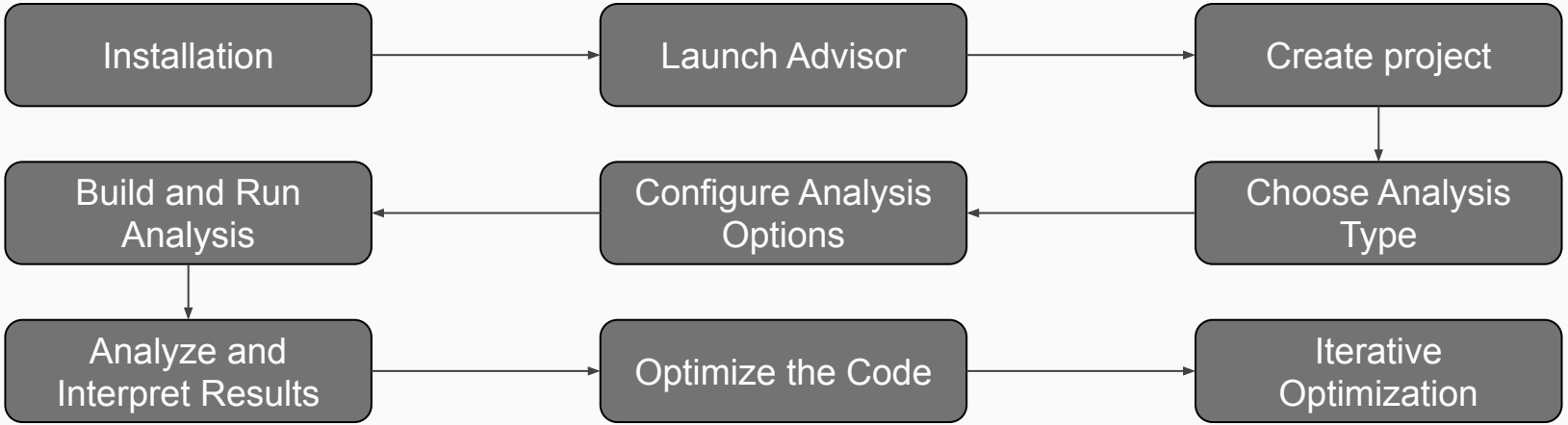
- Intel Advisor is a design and analysis tool for developing performant code.
  Helps you to Design your application for efficient threading, vectorization, and memory use.
  Helps you to identify parts of the code that can be profitably offloaded. Optimize the code for compute and memory.
  Helps you to Create, visualize, analyze computation for heterogeneous algorithms.

Following are some key features and capabilities:

a) Thread Advisor
b) Vectorization Advisor
c) Memory access analysis
d) Roofline analysis
e) Survey and Trip counts
f) Command line and IDE integration

Intel Advisor

- To run Intel Advisor, follow these general steps:



Intel Advisor

- Intel Advisor command line syntax:

```
advisor <--action> [--<action-options>] [--<global-options>] [[--] <target> [<target options>]]
```

-action	: The action to perform, such as collect or report.
-action-option	: Action-options modify behavior specific to the action.
-global-option	: Global-options modify behavior in the same manner for all actions.
target	: The target application to analyse.
target-options	: The options of the application.

More details here:
https://www.intel.com/content/www/us/en/docs/advisor/get-started-guide/2023-1/before-you-begin.html

Intel Advisor

- Following are few popular analysis types available for Intel Advisor:

  - survey
  - tripcounts
  - offload
  - roofline

More details here:
https://www.intel.com/content/dam/develop/external/us/en/documents/advisor-cheat-sheet.pdf

Workout 3

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any FFTW library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for Intel Advisor:

1) Use the application LAMMPS as
the target application for profiling.
2) Execute Intel Advisor for the above listed analysis types.
3) Observe the reports generated.

Intel Inspector

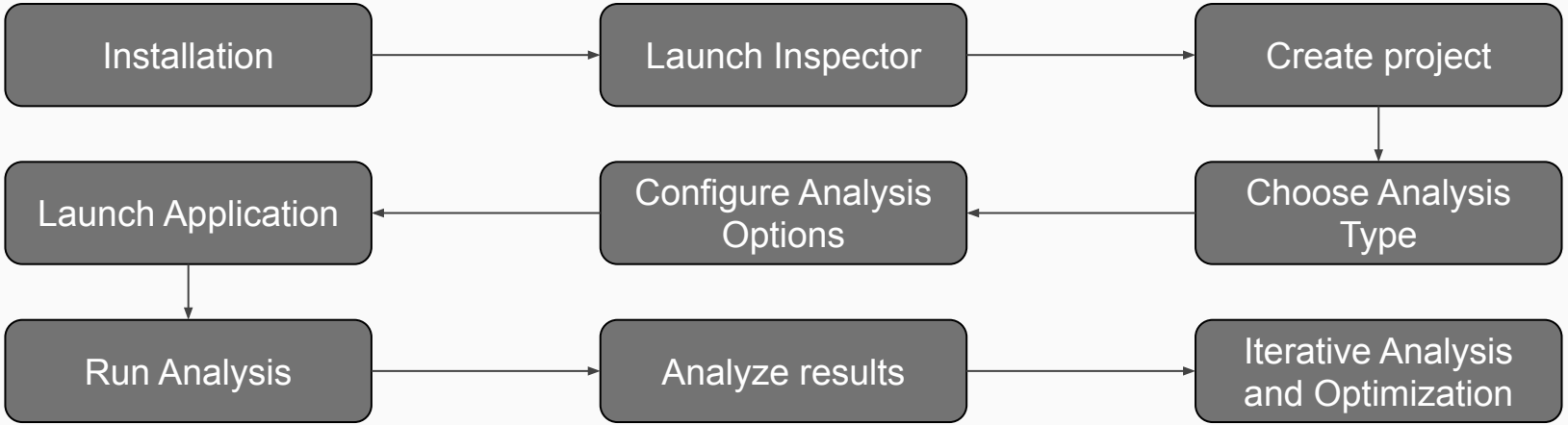
- Memory errors and nondeterministic threading errors are difficult to find without the right tool. Intel Inspector is designed to find these errors.
  Helps locate the root cause of memory, threading, and persistence errors.
  simplifies the diagnosis of difficult errors by breaking into the debugger just before the error occurs
  Check all code, including third-party libraries with unavailable sources

Following are some key features and capabilities:

a) Memory Error Analysis
b) Thread Error Analysis
c) Potential Data Races
d) Memory Leak Detection
e) Detailed Diagnostic Information
f) Performance Impact Analysis

Intel Inspector

- To run Intel Inspector, follow these general steps:



Intel Inspector

- Intel Inspector command line syntax:

```
inspxe-cl <-action> [-action-options] [-global-options] [-- application [app options]]
```

-action	: The action to perform, such as collect or report.
-action-option	: Action-options modify behavior specific to the action.
-global-option	: Global-options modify behavior in the same manner for all actions.
–application	: The target application to analyse.
app-options	: The options of the application.

More details here:
https://www.intel.com/content/www/us/en/docs/inspector/user-guide-linux/2023-1/command-syntax.html

Intel Inspector

- Following are few popular analysis types available for Intel Inspector:

  - Memory Leak Issues
  - Memory Access Issues
  - Data locks
  - Data Races
  - Thread Issues

More details here:
https://www.intel.com/content/www/us/en/docs/inspector/user-guide-linux/2023-1/inspxe-cl-actions-options-and-arguments.html

Workout 4

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any FFTW library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for Intel Inspector:

1) Use the application LAMMPS as
the target application for profiling.
2) Execute Intel Inspector for the above listed analysis types.
3) Observe the reports generated.

HPCToolkit

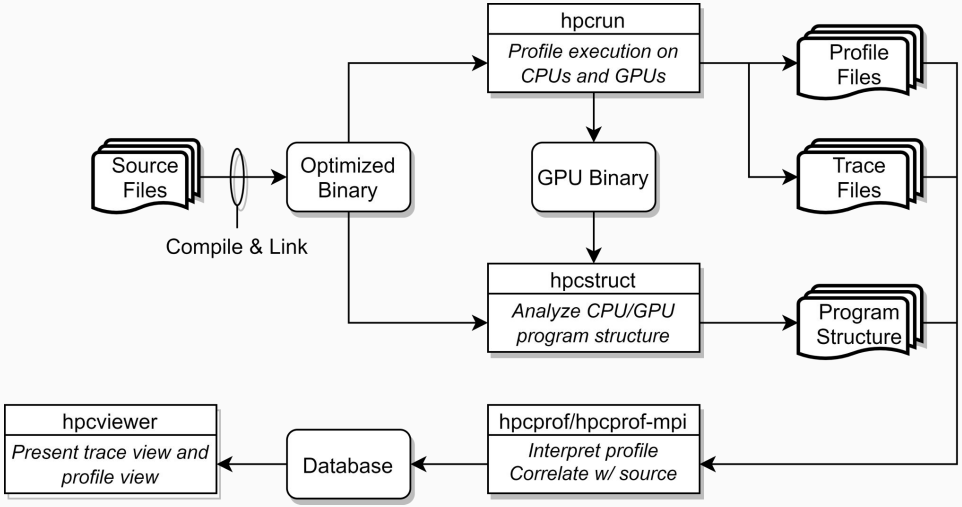
- HPCToolkit is an integrated suite of tools for measurement and analysis of program performance. HPCToolkit works with multilingual, fully optimized applications that are statically or dynamically linked. HPCToolkit supports measurement and analysis of serial codes, threaded codes (e.g. pthreads, OpenMP), MPI, and hybrid (MPI+threads) parallel codes, as well as GPU-accelerated codes that offload computation to AMD, Intel, or NVIDIA GPUs.

Following are some key features and capabilities:

a) Binary Level Measurement and Analysis
b) Scalability
c) Application Coverage
d) Call Path Profiling
e) Visualizations and Reports
f) Open-Source and Community-Driven
g) Supports top-down performance analysis

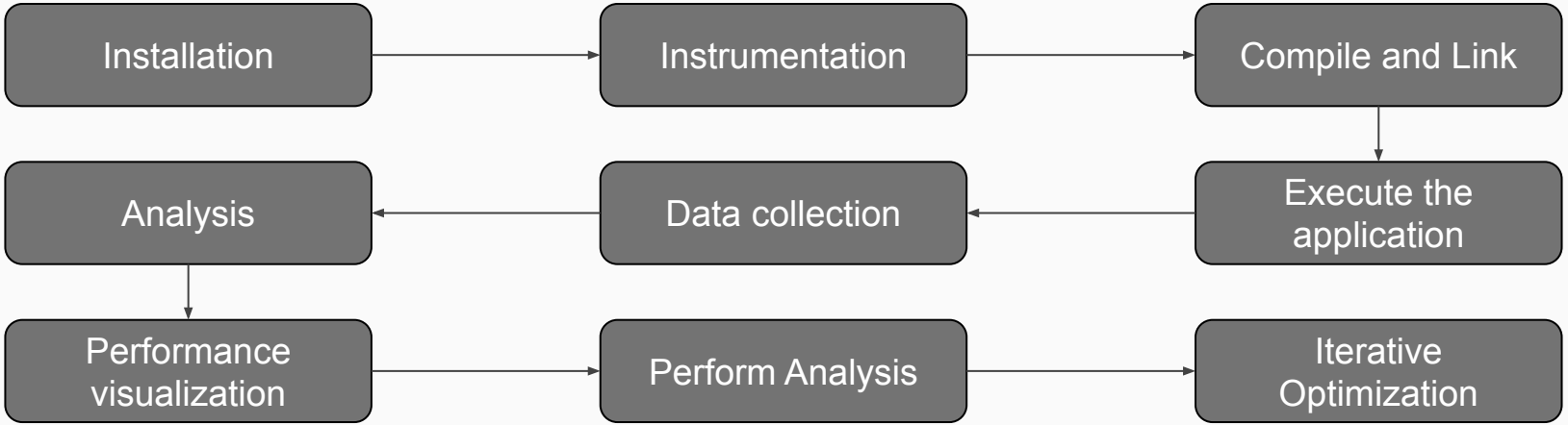
HPCToolkit

- The figure below illustrates HPCToolkit's primary components and their relationships.



HPCToolkit

- To run HPCToolkit, follow these general steps:



HPCToolkit

- HPCToolkit command syntax (Measurement):

```
[<mpi-launcher>]  hpcrun  [hpcrun-options]  app  [app-arguments]
```

The command will produce a measurements database that contains separate measurement information for each MPI rank and thread in the application.
The database is named according the form:

```
hpctoolkit-app-measurements[-jobid]
```

HPCToolkit

- HPCToolkit command syntax (Analysis):

  To analyze HPCToolkit's measurements and attribute them to the application's source code, use hpcprof, typically invoked as follows:

```
[<mpi-launcher>]  <mpi-params>  hpcprof-mpi  hpctoolkit-app-measurements
```

HPCToolkit

- HPCToolkit command syntax (Analysis):

  To analyze HPCToolkit's measurements and attribute them to the application's source code, use hpcprof, typically invoked as follows:

```
[<mpi-launcher>]  <mpi-params>  hpcprof-mpi  hpctoolkit-app-measurements
```

- To interactively view and analyze an HPCToolkit performance database, use hpcviewer.

```
hpcviewer  hpctoolkit-app-database
```

HPCToolkit

- Following are few popular collection types available for HPCToolkit:

  - Hardware event collection (-e)
  - Trace generation (-t)
  - Delay sampling (-ds)
  - Disable sampling

More details here:
http://hpctoolkit.org/man/hpcrun.html

Workout 5

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any MPI library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for HPC Toolkit:

1) Use the application LAMMPS as
the target application for profiling.
2) Execute hpcrun and hpcprof binaries successfully.
3) Observe the reports generated.

TAU (Tuning and Analysis Utilities)

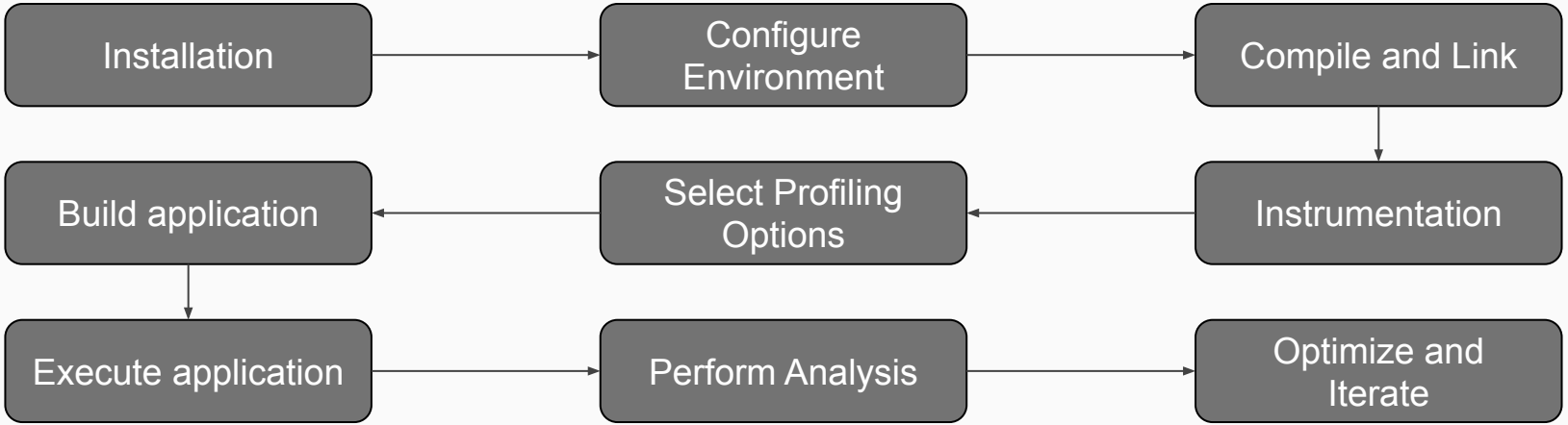
- TAU Performance System is a portable profiling and tracing toolkit for performance analysis of parallel programs.
  TAU (Tuning and Analysis Utilities) is capable of gathering performance information through instrumentation of functions, methods, basic blocks, and statements as well as event-based sampling. The API also provides selection of profiling groups for organizing and controlling instrumentation.

Following are some key features and capabilities:

a) Profiling Capabilities
b) Performance Measurement
c) Instrumentation
d) Scalability
e) Measurement Overhead Control
f) Multi-Language Support
g) Analysis and Visualization

TAU

- To run TAU, follow these general steps:



TAU

- To perform compiler based **instrumentation** using TAU:

```
tau_cc.sh    -tau_options=-optCompInst    samplecprogram.c
```

A list of options for the TAU compiler scripts can be found by typing:

```
man    tau_compiler.sh
```

TAU

- After instrumentation and compilation are completed,
  the profiled application is **run** to generate the **profile data** files.

```
export TAU_VERBOSE=1;  mpirun  -np  4  samplecprogram.out
```

You can enable **callpath** profiling by setting the environment variable TAU_CALLPATH.

```
export  TAU_CALLPATH=1
```

TAU

- To enable **tracing** with TAU, set the environment variable TAU_TRACE to 1.

```
export TAU_TRACE=1;
```

This will generate a trace file and an event file for each processor.
To merge these files, use:

```
tau_treemerge.pl
```

TAU

- For a quick **text based view** summary of TAU performance, use

```
pprof
```

- You can also use ‘ParaProf’ which will display the profile data on GUI.

```
paraprof
```

- To view trace information, you can use the jumpshot tool.

```
#Conversion from TAU trace to slog2 files => tau2slog2 tau.trc tau.edf -o tau.slog2
jumpshot   tau.slog
```

TAU

- For a Quick Reference guide:
  https://www.cs.uoregon.edu/research/tau/docs/newguide/bk01pt01ch05.html
- For User Guide:
  https://www.cs.uoregon.edu/research/tau/docs/newguide/index.html
- Additional tools that can be used with TAU:
  Vampir, PerfExplorer, Paraver
- TAU installation guide:
  https://www.cs.uoregon.edu/research/tau/docs/newguide/bk02ch01.html

Workout 6

A) Currently, we are proceeding with the GNU versions of the compilers, try and check if any different behaviour is observed when using Intel compilers.

B) Do you observe any MPI library calls ?

C) List the most time consuming operations that are observed.

D) Identify the type of bottleneck that the application has. (Compute, I/O, Memory, Communication, etc)

Follow the steps listed for TAU using instrumentation and tracing:

1) Use the application LAMMPS as
the target application for profiling.
2) Compile the application using TAU compilers
and display the profile using pprof
3) Observe the reports generated.

“Profiling tools are like detectives for your code—they catch the performance culprits red-handed, but sometimes they need a little help deciphering their statements. Optimize wisely, and together, we'll crack the case of the sluggish software!”

Questions?

Thanks!

Contact:

Vineet More
vineet.more.bfab@gmail.com
[Make sure to use HPCAP23 as the subject line for your queries]

LinkedIn Handle:
https://www.linkedin.com/in/vineet-more-c-programmer/