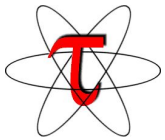
A Plugin Infrastructure for MPI Performance Engineering: Integrating MVAPICH and TAU via the MPI tools interface

Srinivasan Ramesh, Sameer Shende, Aurele Maheo, Allen. D. Malony
University of Oregon

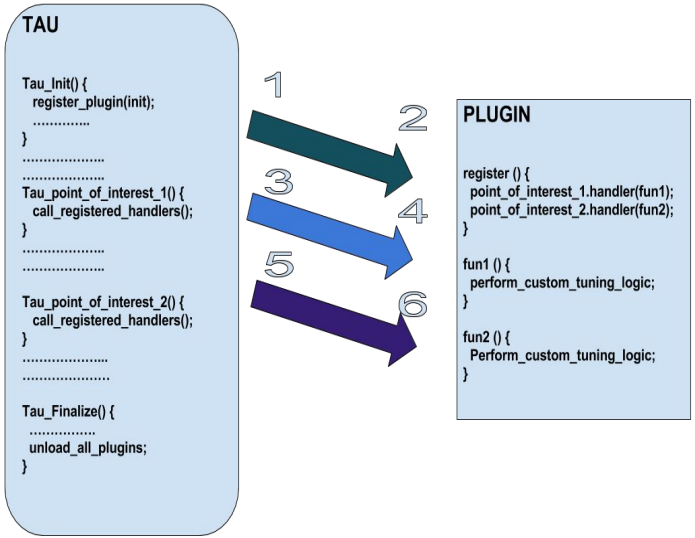
Hari Subramoni, Dhabaleswar. K. Panda
The Ohio State University



ABSTRACT:
MPI libraries today are complex software's that have modular components reflecting the constantly evolving software and hardware stack underneath. They offer the user a number of tunable parameters that can affect performance. In this environment, performance variations and scalability limitations can come from a number of sources. The MPI Tools Information Interface (MPI_T) introduced as part of the MPI 3.0 standard offers a standardized mechanism through which external tools can interact with an MPI library to perform runtime introspection and performance tuning. In this work, we describe a plugin infrastructure within TAU that utilizes the MPI_T interface to perform runtime tuning of MPI libraries, and generates performance recommendations to the user.

Plugin Infrastructure in TAU

- Plugins infrastructures are effective in ensuring that custom logic within a software tool is maintained properly, and offer a clean method of adding extensions to the software tool
- TAU is a popular profiling software that has capabilities to query the MPI Tools interface at runtime, and re-configure the MPI library dynamically through the MPI Tools interface
- To design such capabilities, the following need to be considered:
  - **How does one handle custom MPI_T based tuning logic that varies between implementations?**
  - **How do we offer an expert user the capability to write custom logic without re-writing the main software component every time?**
- A natural solution is to develop a plugin infrastructure within TAU that allows the user to write custom MPI_T based tuning plugins, and load only the necessary tuning plugins specified through the command line
  - Each plugin registers handlers (functions) for one or more “**well-defined” “points of interest”** inside TAU such as:
    - TAU initialize
    - TAU finalize (end of profiling run)
    - Event-based triggers (interrupts)
  - At each of these “points of interest”, TAU invokes registered handlers of plugins that offer that specific functionality



Case Study: Autotuning AmberMD and 3DStencil

- AmberMD is a molecular dynamics application that uses asynchronous point-to-point routines with message sizes in the ~ 64KB - 200KB range
- 3DStencil is a synthetic stencil application we developed that performs point-point communication of fixed message size
- Significant performance gains visible with a higher value for Eager threshold for both AmberMD and 3DStencil applications
  - Disadvantage of high Eager threshold: Larger memory footprint of MPI
- When profiling with TAU through the MPI_T interface, we notice that virtual buffers (VBUFs) from all pools except one remain largely unused.
  - **Tuning Opportunity: Free unused VBUFs from pools that aren't utilized**
- **Autotuning plugin: Reads number of unused VBUFs through MPI_T and sets MVAPICH2 CVAR enabling freeing of unused VBUFs if too many unused VBUFs exist in a pool**

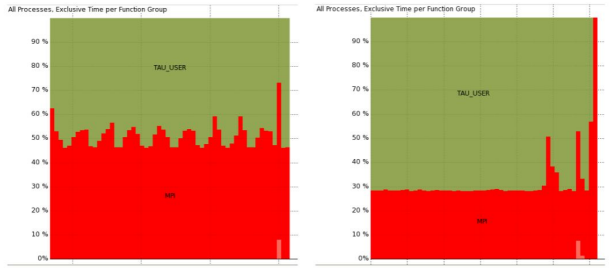
Results of applying autotuning to AmberMD and 3DStencil

Table 1: Amber - Impact of Eager threshold and autotuning on execution time and memory usage

Run	Number of Processes	Eager Threshold	MD Timesteps	Application Runtime(Seconds)	Total VBUF Memory(Bytes)
Default	128	MVAPICH2 Default	4000	210	695150
Eager	128	64000	4000	129	768188
TAU runtime autotuning	128	64000	4000	129	629511

Table 2: 3DStencil - Impact of Eager threshold and autotuning on execution time and memory usage

Run	Number of Processes	Message Size(Bytes)	Communication-Computation Overlap	Eager Threshold	Total VBUF Memory(Bytes)
Default	448	32,768	11.1	MVAPICH2 Default	1436574
Eager	448	32,768	68.7	33000	2573345
TAU runtime tuning	448	32,768	69.7	33000	1208782



(a) Before Eager threshold tuning (b) After Eager threshold tuning

Figure 4: Vampir[22] summary process timeline view of 3DStencil shows increased time in user code after runtime optimization of communication

Case Study: Recommending Hardware Offloading

- Implementations such as OpenMPI export 1000+ CVARs - the user may not have the knowledge to find the right CVAR to tune
- In such a situation, TAU can be used to generate performance recommendations / hints to the user based on the information collected during the profiling run from both PMPI and MPI_T interfaces
- **We can re-use the plugin infrastructure for this purpose - the recommendation policy is a plugin that “hooks into” TAU**
- MiniAMR (a mini-app from Sandia) uses MPI_Allreduce with a message size of 8 bytes for checksumming routine (latency sensitive)
  - Prime candidate to benefit from hardware offloading using SHArP hardware offload protocol supported by MVAPICH2 for MPI_Allreduce operation
- **Recommendation Plugin:**
  - Plugin monitors message size through PMPI interface
  - If message size is low, and execution time inside MPI_Allreduce is significant, a recommendation is generated on ParaProf (TAU’s GUI) for the user to set the CVAR enabling SHArP

Generating Performance Recommendations through TAU

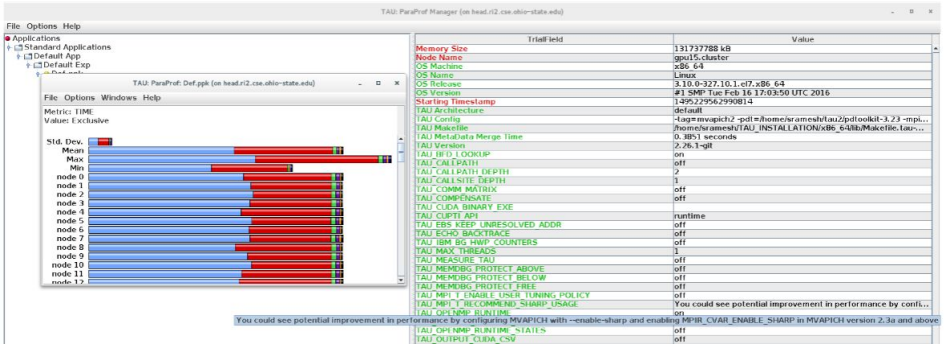


Table 3: MiniAMR - Impact of hardware offloading on application runtime

Run	Number of Processes	Total Application Runtime (secs)
Default	224	648
SHArP enabled	224	618

Future Work

- How can we support the specification of plugin autotuning policies in a generic fashion?
  - User shouldn’t be writing MPI-specific code for tuning policies
  - Interface should allow easy customizability of tuning policy (for changing thresholds, etc) without needing to re-compiling the tuning logic
- Develop a deeper understanding of impact of CVARs on performance of applications on large scale
  - Build intelligence into TAU for autotuning a wide variety of applications and patterns through MPI_T
- Provide the user with a more interactive user-interface for performance tuning
- Enrich MVAPICH2 to support more applications and use cases for MPI_T runtime tuning