Adaptive auto-tuning in HPX using APEX

MOHAMMAD ALAUL HAQUE MONIL, University of Oregon
BIBEK WAGLE, Louisiana State University
KEVIN HUCK, Performance Research Lab, University of Oregon
HARTMUT KAISER, Louisiana State University

ABSTRACT

Finding an optimal value for a parameter that impacts application performance is a hard problem and often requires repetitive execution of the application. This auto-tuning effort by running the application many times causes wastage of HPC resources. In this research, we provide a preliminary study which demonstrates parameter value searching at runtime while ensuring better performance. We use APEX performance measurement library to implement an adaptive auto-tuning policy to tune parcel coalescing parameters based on a sampled counter of HPX runtime system. Our study shows APEX policy can converge on parcel coalescing parameters while reducing the network overhead and hence ensures reduction of execution time.

KEYWORDS

Task-Based Runtime, APEX, HPX, auto-tuning.

1 EXTENDED ABSTRACT

Being on the verge of exascale era, researchers are trying to come up with novel programming model and runtimes to better utilize the HPC resources. Task-based programming model and runtime is one of the happening areas in the field High-Performance Computing(HPC). During the last decade, a good number of such programming model and runtimes are made available by the HPC research community. Some of the runtime systems provide asynchronous execution which is very different from tradition MPI programming model. This difference opens the door for new research to understand and look at application performance from a new perspective.

In this research, we worked on APEX (Autonomic Performance Environment for Exascale), a performance measurement library for distributed, asynchronous tasking models/runtimes. i.e. HPX. It provides lightweight measurement (task < 1ms) and high concurrency. To support performance measurement in asynchronous task-based runtime systems, it depends on dependency chain rather call stack. HPX is such an asynchronous task-based runtime system which builds on C++ runtime system based on the ParalleX model. The HPX threading system employs lightweight tasks, known as HPX threads, that are scheduled on top of operating system threads. In a distributed environment, a locality in HPX is an abstraction for a physical node. The Active Global Address Space (AGAS) system in HPX provides a mechanism for addressing any HPX object globally. In this study, we consider improving parcel coalescing parameters at runtime for HPX runtime system using APEX policy engine.

Finding an optimal value for a parameter that impacts application performance is a hard problem and often requires repetitive execution and hence incurs wastage of resources. In this poster, we provide a preliminary study which demonstrates parameter value searching at runtime for better performance. We implement an adaptive auto-tuning policy to tune parcel coalescing parameters based on a sampled counter of HPX runtime system. Following previous research, where a direct correlation between network overhead and application execution time was shown

provided a solid ground to develop an adaptive policy in APEX policy engine. The main objective of this policy is to find out optimal value for two parameters: interval between the coalesced message sent and the number message to have coalesced. The APEX custom policy triggers the policy based on specified interval or based on the specified number of send events. When the policy is triggered, it invokes active harmony auto tuning library. Active harmony observes the sampled network overhead counter and makes an adjustment to the values of the parameters. If the overhead is reduced in the next iteration, more change is made in that direction and if no improvement is found the parameters are changed in the opposite direction. For every policy triggering event, the callback function ensures the parameter values go closer to better performance. In this way, the policy search for parameter values for which the application demonstrates better performance based on the selected counter from the runtime system. Finding the counter that correlates with application performance and when to trigger the policy warrant research effort. While in this poster we demonstrate the counter correlation we still need to find when is the optimal time to trigger the policy.

Preliminary results based on toy application showed promising result and provided the proof of concept. However, we plan to verify our results by running real application at scale.