Monitoring Clusters Using Extended Berkeley Packet Filter (eBPF)

Weston Cadena (westoncadena@tamu.edu); Alexis Ng (alexism.ng@ku.edu); Matthew Phillips (mphillips8687@gmail.com) | Mentors: Travis Cotton, Nick Jones, Johnathan Nielsen

Background

eBPF Overview

eBPF tools are used for **performance analysis and observability** at a low overhead and cost by attaching probes (kernel, user, tracepoints). This project primarily focused on the BPF Compiler Collection (BCC) tools: pre-written tools utilizing eBPF. They provide greater insight into the processes that are running "under the hood" of the Linux kernel and software applications using tracing, sampling, and snooping.

eBPF in HPC

eBPF can target certain layers within the cluster (filesystems, CPU usage, network connections, etc.), which are typically blackboxed when diagnosing problems and monitoring the system beyond the existing Linux tools. eBPF output can also be written directly to /var/logs or into files.

Admins can use eBPF

- across separate nodes to diagnose slow or congested nodes
- diagnose user issues and cluster problems using kernel probes
- create custom probes based on syscalls, bringing flexibility and relevance in HPC for accurate targeting

Users can use eBPF to

- find inefficiencies in their code
- understand how their code scales and distributes work during jobs.
- with root access, but without it, system admins can find alternative solutions to grant permissions for desired tools.

Project Scope

We aimed to determine the viability of BCC tools in an HPC environment by monitoring a 10 node cluster while performing kernel compilation and physics simulations testing. Tool output, obtained through a series of benchmarking tests, was graphed using R to evaluate the clarity and efficacy of the tool results.

Conclusion

eBPF was able to provide reasoning and insight into slow nodes, high levels of traffic, and CPU activity at a detailed level. With each tool having different purposes of debugging or job evaluations, eBPF is able to help scientists and admins reach performance objectives.

Future Work

Greater testing on overhead and integration within the existing ecosystems for logging softwares (Splunk, Grafana) is recommended.

Methods

A systemd service was used to run the BCC's tools to monitor our desired benchmarks. The service is able to start up tools, redirect outputs, and the end them correctly with a sigint-preventing corrupted or incomplete outputs.

Vector Particle-In-Cell (VPIC)

A physics code simulating coupled Maxwell-Boltzmann system of equations. VPIC uses Message Passage Interface (MPI) calls for multi-node applications. In addition, VPIC checkpoints its data, allowing for flexible restarts at the cost of excess writes during the simulation.

bpftrace

`bpftrace` is a front end tool meant for one-liners and custom scripts allowing for flexibility and more specific probing on the stack. `bpftrace` does not allow for custom probes; however, the pre-made probes target syscalls much more effectively than BCC tools. https://github.com/iovisor/bpftrace

TCP

Context/Purpose: Sampling the TCP transmissions of ***VPIC Harris*** input deck using **tcptop** without mpi working (figure 1) and with MPI working (figure 2) amongst nodes. The width is based on the amount of data transmitted in kilobytes between processes and the color is based off process.

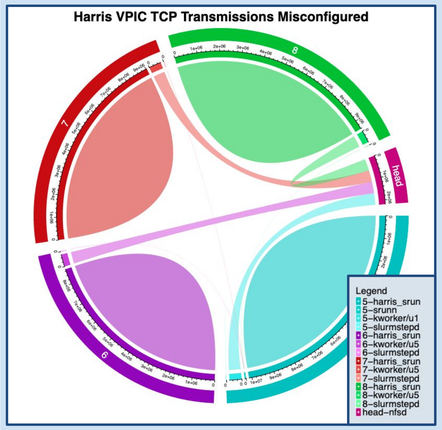

Figure 1: TCP Transmissions without MPI

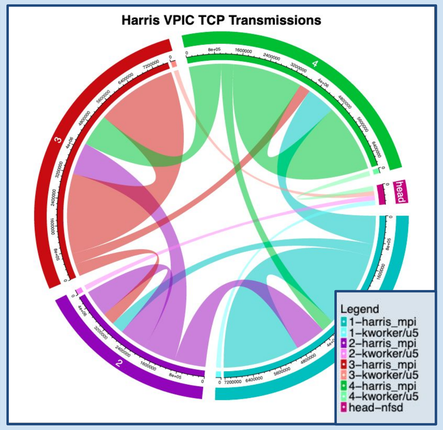

Figure 2: TCP Transmissions with MPI

CPU

Context/Purpose: Tracking user stack with the **offcputime** tool on ***VPIC-KOKKOS benchmark*** **test.** Figure 3 shows the current load on the system from the test. It may be noticed that there is "[unknown]" which is when the user stack can not recursively find the process name.

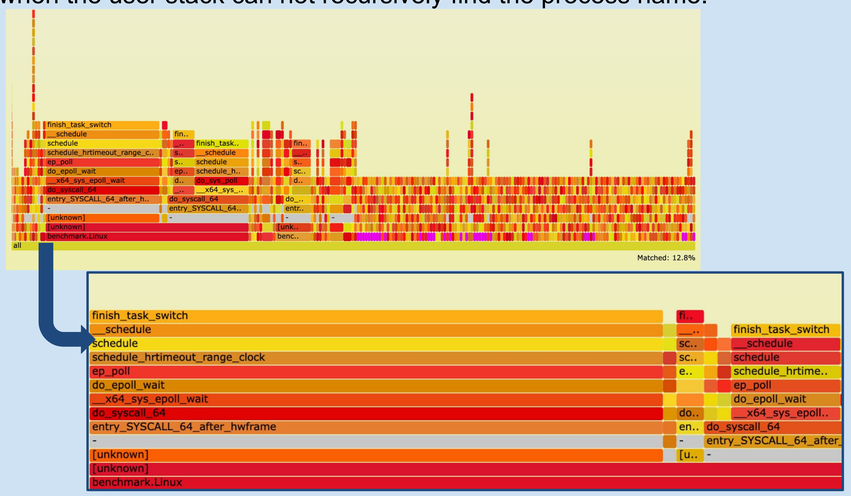

Figure 3: CPU User Stack Visualization

Memory

Context/Purpose: Sampling the CPU cache hits and TCP transmission amounts with **llcstat** and **tcptop** every second on ***VPIC-KOKKOS benchmark*** test. Figure 4 shows ~20 minute snapshot of a 2 hour test.

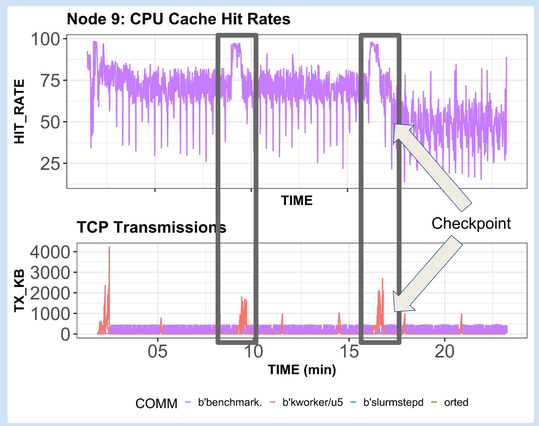

Figure 4: CPU Cache and TCP Transmission over time

File Systems

Context/Purpose: Tracing the reads and writes of three serial ***kernel compilation*** using sbatch to stress the NFS. BCC's **nfsslower** shows reads/writes with latencies greater than 1 ms. Figure 5 shows a scatter plot of reads and writes per command separated by node. Figure 6 shows that there are outliers of the xz command (data decompression).

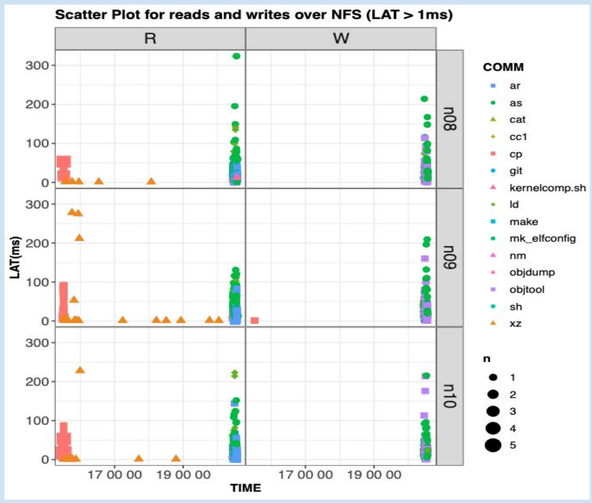

Figure 5: Read/Write on NFS over Time

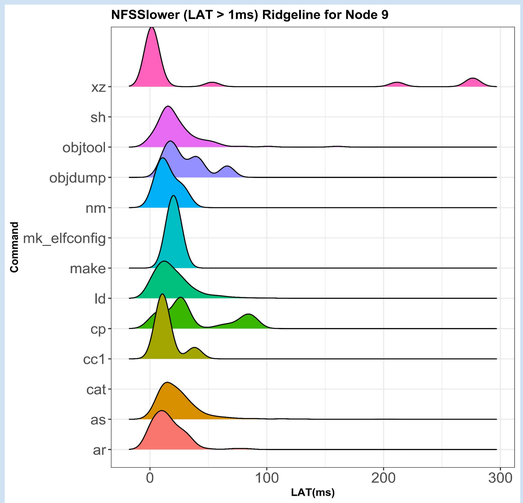

Figure 6: Command Latency on Node 9

Los Alamos NATIONAL LABORATORY
National Nuclear Security Administration
Managed by Triad National Security, LLC, for the U.S. Department of Energy's NNSA.
LA-UR-23-28968
eBPF
TEXAS A&M UNIVERSITY
KU THE UNIVERSITY OF KANSAS
MONTANA STATE UNIVERSITY