GWDG
WG Computing
Jack Ogaja

Tutorial 1 / 21.04.2023
Practical Course in Parallel Computing / SoSe 2022/23
50 Minutes Total

Node-Level Performance Analysis with VAMPIR

Learning Objective

The objective of this tutorial is:

- To learn how to use VAMPIR performance tools to explore and understand performance characteristics of a matrix multiplication algorithm
- To learn how to develop an Empirical Performance Model and identify optimization options of common computational kernels.

Tools

- VAMPIR

Contents

1 Introduction

In this tutorial and exercises we use VAMPIR performance tool-suite to evaluate the performance of a specific computational kernel. The included source code does matrix multiplication using Cannon's algorithm - designed to improve memory efficiency. The exercise is to instrument the code, generate trace files using Score-P and visualize the trace files using VAMPIR to learn about the performance characteristics of the algorithm on the available hardware. You are also expected to use customized performance metrics e.g. (FLOPS and L3_TCM) to generate a Roofline model, identify bottlenecks and computation and bandwidth boundedness of the algorithm. Suggest Optimization strategy based on an Empirical Roofline Model.

1.1 Cannon's Algorithm

Cannon's Algorithm

// make initial alignment
for i, j :=0 **to** $\sqrt{p} - 1$ **do**
Send block $A_{i,j}$ to process $\left(i, \left(j - i + \sqrt{p}\right) mod \sqrt{p}\right)$ and block $B_{i,j}$ to process $\left(\left(i - j + \sqrt{p}\right) mod \sqrt{p}, j\right)$;
endfor;
Process $P_{i,j}$ multiply received submatrices together and add the result to $C_{i,j}$;

// compute-and-shift. A sequence of one-step shifts pairs up $A_{i,k}$ and $B_{k,j}$
// on process $P_{i,j}$. $C_{i,j} = C_{i,j} + A_{i,k} B_{k,j}$
for step :=1 **to** $\sqrt{p} - 1$ **do**
Shift $A_{i,j}$ one step left (with wraparound) and $B_{i,j}$ one step up (with wraparound);
Process $P_{i,j}$ multiply received submatrices together and add the result to $C_{i,j}$;
Endfor;

NOTE: Tasks for this tutorial and exercises should be performed in compute nodes of GWDG's Scientific Compute Cluster (SCC).

Use VAMPIR to visualize trace data 2: Tutorial (15 min)

Use VAMPIR tool-suite to generate and explore trace data of the given source code to characterize the performance of Cannon's algorithm.

Steps

1. Compile and instrument the source code `main.c`
2. Generate trace data from the instrumented code
3. Explore the Master Timeline chart from the trace data
4. Explore the Process Timeline chart from the trace data
5. Explore the Counter Data Timeline chart from the trace data
6. Explore the Performance Radar from the trace data
7. Customize the performance metric, `Wait_time`
8. Customize the performance metric, `FLOPS`
9. Explore the memory allocation
10. Explore the function and process summaries

11. Explore the communication matrix and identify communication imbalances if any.

Hints

- Use the provided shell scripts, `compile_instrument.sh` and `run_trace.sh`.

Empirical Roofline Model 3: Tutorial (35 min)

Use customized performance metrics to measure the performance of the given program. Is the program computation or communication bound? Can you identify any bottleneck? Are there possible additional optimization strategy?

Steps

1. Compile and instrument the given source code
2. Customize and measure performance metrics `FP_FLOPS` and `L3_TBW` - L3 bandwidth to generate an empirical Roofline Model
3. Compare the measured performance to vendor's specifications.

Hints

- Use the provided shell scripts, `compile_instrument.sh` and `run_trace.sh`.

Further Reading

- VAMPIR Performance Tools - https://vampir.eu/tutorial/manual
- Samuel W. Williams (2008) Auto-tuning Performance on Multicore Computers, *University of California at Berkeley, Technical Report No. UCB/EECS-2008-16*