Mitos: A Simple Interface for Complex Hardware Sampling and Attribution

Alfredo Giménez[1,2], Benafsh Husain[1], David Böhme[1], Todd Gamblin[1], Martin Schulz[1]

*Lawrence Livermore National Laboratories, University of California, Davis

Abstract

As high-performance architectures become more intricate and complex, so too do the capabilities to monitor their performance. In particular, hardware sampling mechanisms have extended beyond the traditional scope of code profiling to include performance data relevant to the data address space, hardware topology, and load latency. However, these new mechanisms do not adhere to a common specification and as such, require architecture-specific knowledge and setup to properly function. In addition, the incoming data is often low-level and difficult to attribute to any useful context without a high level of expertise. This has resulted in a destructively steep barrier to entry, both in data acquisition and analysis.

Mitos provides a simple interface for modern hardware sampling mechanisms and the ability to define attributions of low-level samples to intuitive contexts. We present the Mitos API and demonstrate how to use it to create detailed yet understandable visualizations of performance data for analysis.

Mitos Overview

From the user perspective, using Mitos involves:

1. A sample handler function

```
// Mitos output structure
Mitos_output mout;

// Mitos sample handler function
void sample_handler(perf_event_sample *sample, void *args)
{
    // Write to output using Mitos API
    Mitos_write_sample(sample, &mout);
}
```

2. Setup and configuration

```
    Mitos_create_output(&mout);
    Mitos_pre_process(&mout);
    Mitos_set_sample_latency_threshold(10);
    Mitos_set_sample_time_period(4000);
```

2. Annotating data objects

```
    // Allocate data structures
    double *a = new double[N*N];

    // Annotate data structures
    size_t dims[2] = {N,N};
    Mitos_add_symbol("a",*a,sizeof(double),dims,2);
```

4. Calling begin/end sampling

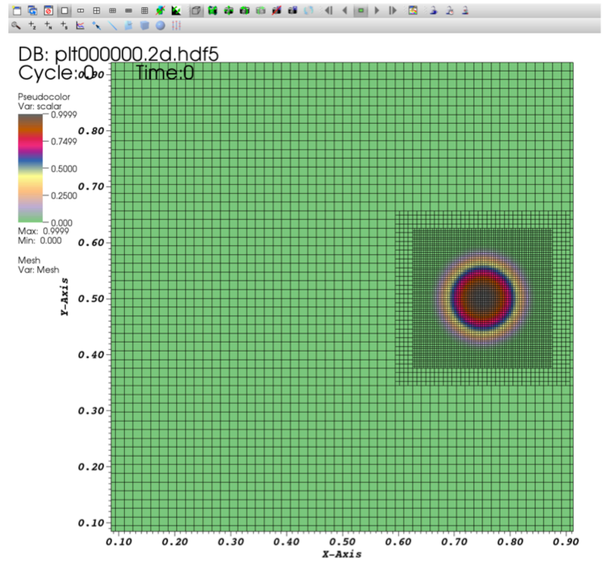
```
    Mitos_begin_sampling();
    ...
    Mitos_end_sampling();
    Mitos_post_process(&mout);
```

This will generate ~**4000 samples per second**, for loads that take **>10 cycles**, and output them in Mitos' predefined format **Mitos_output**.

It is not necessary to use this particular output format and sample handler, so the user may customize the handler to their needs.

Example Application

We used Mitos to sample data accesses on a simple adaptive mesh refinement (AMR) code. The code produces the following simulation output:



Data Access Sampling w/Mitos

Raw Samples

The data produced for each sampled data access before performing any additional attribution includes:

1. Instruction Pointer
2. Core ID
3. Process ID
4. Thread ID
5. Timestamp
6. Loaded Data Address
7. Instruction Latency
8. Data Source Encoding (L1, L2, RAM, Remote RAM)

While this is highly detailed, it may be difficult to interpret without the context of the code, data objects, hardware topology, or physical problem domain.

ip	pid	tid	time	addr	cpu	latency	data_src
4731100	236105	236105	1.23442E+15	10927424	0	11	1745879362
4731095	236105	236105	1.23442E+15	10926544	0	11	1745879362
4731084	236105	236105	1.23442E+15	10926936	0	8	1745879362
4731100	236105	236105	1.23442E+15	10927592	0	11	1745879362
4731084	236105	236105	1.23442E+15	10931120	0	8	1745879362
4731100	236105	236105	1.23442E+15	10932112	0	12	1745879362
5612623	236105	236105	1.23442E+15	11701832	0	13	1745879362
4731109	236105	236105	1.23442E+15	11273648	0	8	1745879362
4731084	236105	236105	1.23442E+15	9718968	0	12	1745879362
4731095	236105	236105	1.23442E+15	9717376	0	12	1745879362
5619000	236105	236105	1.23442E+15	10278352	0	14	1745879618
4718603	236105	236105	1.23443E+15	9722088	0	12	1745879362
4731100	236105	236105	1.23443E+15	9715952	0	8	1745879362

Mitos Attribution

Source Code

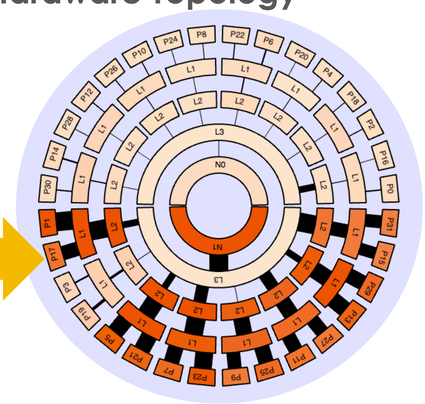
```
AMRBox.cpp : line 42
double v = matrix_1[x][y];
```

Data Object and Access Index

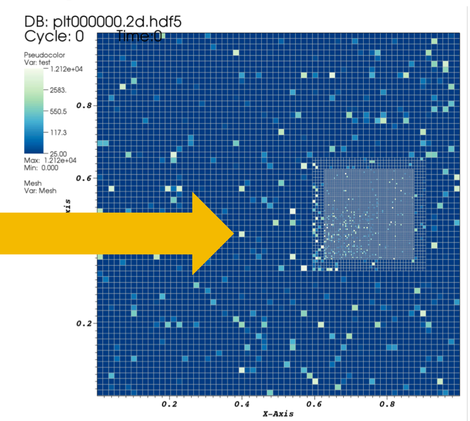
matrix_1

0,0	1,0	2,0	3,0	4,0
		2,1	3,1	4,1
0,2	1,2	2,2	3,2	4,2
0,2	1,3	2,3	3,3	4,3

Hardware Topology



Physical Coordinates



github.com/scalability-llnl/Mitos
github.com/scalability-llnl/MemAxes

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and supported by Office of Science, Office of Advanced Scientific Computing Research as well as the Advanced Simulation and Computing (ASC) program.