Project Title: saveCHIMP (save CHIp Multi Processor)

Team Name: DAT

Team Members: Arjun Khurana, Dong-hyeon Park, Timothy Wong

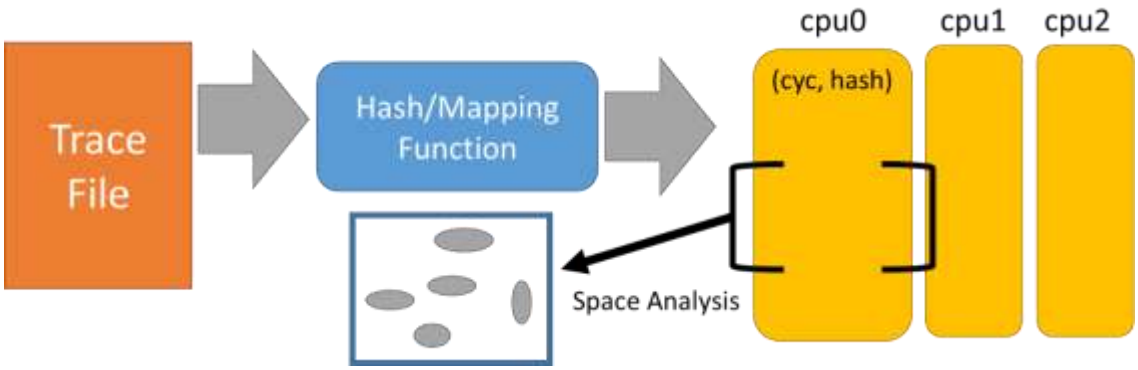
Idea/solution to be investigated by the project

We want a framework to do the following:

1. Given a program/software expected to run on the system, identify the characteristic transactions and behaviors that are exercised by the program.
2. Represent the transactions/behaviors in a model that can help us identify a search space.
3. Generate several series of targeted tests for the particular search space we are interested in. We will focus our effort in this project on applying our idea on a homogeneous system (e.g. a network of CPUs).

Details on Behavioral Analysis:

1) Given the trace, use a hash/mapping function to map the instruction or network state to a single value. The function needs to be able to capture the information we are interested in (type of memory operation, address, destination address) while not losing locality of these parameters.
2) Using the function, we breakdown the trace files to a sequence of states and separate them by CPU.
3) Create a transition table of the states for the each segment, to represent the behavior.



Progress so Far

Since last checkpoint we:

- Designed a hashing/mapping function for the instruction trace that maps the details of memory operation to a search space.
- Organized data collected by hashing memory reads and writes into different states, depending on the registers and addresses read from and written to. We converted the trace file to a sequence of hashed state values for each CPU.
- We tried to represent the relationships between the states by constructing a transition table that shows the frequency of transition from one state to another. This only retains a single transition history, but when we visualized the data, we were able to identify some chains of frequent transitions.

- Sample data:
    - In the diagrams below, the left and middle diagrams show the memory state transition matrices of Core 0 on two different segments of the Blackscholes benchmark, and the right diagram shows the memory state transition matrices of Cores 1-15 in the Blackscholes benchmark.
    - Each point on a matrix represents the transition from one state to another. For example, the point (2,75) represents the transition from state 2 to state 75. The state indices are determined by a hash function.
    - As observed from the left and middle matrices, Core 0 is responsible for running the main scheduling threads, and hence has a significantly larger amount of activity than Cores 1-15. From the right matrix, we can see that Cores 1-15 have a very monotonous pattern of state transitions.

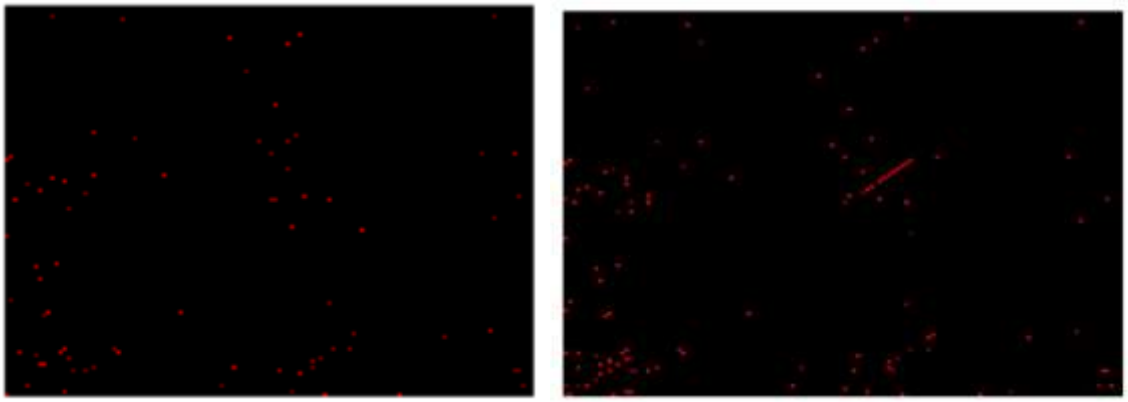
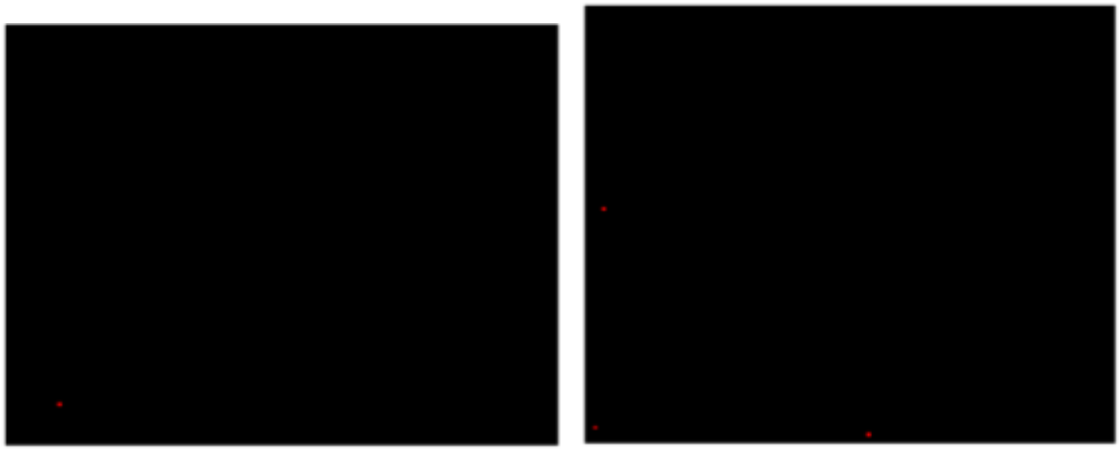


Figure 1-2: Blackscholes Core 1 Segment 1 (left) Core 1 Segment 2 (right)



Blackscholes Core 2-15 (left) Swaptions Core 1 (right)

Issues/Showstoppers

Main challenges we are currently facing:

- We are not confident whether the abstractions (hash functions, the transition model) were relevant to our goal.
- The size of the debug and trace files are extremely large and slowing down our analysis. We are working with a highly condensed dataset, but even then the file size is extremely large.