Midterm Project

CSCI 560

Dr. Paul West

Jordan Manier

Gem5 is a full simulation computer architecture. This system components can be simply replaced rearranged and parameterized to suit the needs of the simulation. Gem5 (m5) was designed to simulate the passing of time as a series of discrete events. Gem5 was written in both C++ and python. Gem5 can either be used in FS mode (full system mode) or SE mode (syscall emulation mode). FS mode allows for the a complete simulation with devices and an operating system. While SE mode will allow for the use of user space only programs. system services are provided directly by the simulator.

The configurations in the first architecture is a simple two core system. A simple 2 core system allows for two L1 caches to be utilized in each core. This architecture isn’t the most ideal for performance in instruction execution. Each of the cores are OOO (out-of-order) this allows for the instructions to be fetched in compiler-generated order (O3) at 1 GHz. I chose to add (attempt to add) 3GB of memory to the architecture. This second architecture is similar to the first. O3 cores operating at 1GHz, with a L1 cache of 128kB. Adding 2 more cores making it a simple 4 core system allows for more instructions to be ran and performance to be enhanced. This architecture I also added 3GB of memory to produce better/faster results. The complex 4 core system adds 2 more caches, a level 2 and 3. Adding more caches allow for better performance and higher hit rate when executing instructions.

1. A simple two core system:

(a) Each core is Out-Of-Order (OOO) at 1 GHZ.

(b) Each core has a L1 cache of size 128kb.

(c) The L1 cache is connected to main memory, with is DDR3 1600.

(d) No other caches.

2. A simple 4 core system:

(a) Each core is Out-Of-Order (OOO) at 1 GHZ.

(b) Each core has a L1 cache of size 128kb.

(c) The L1 cache is connected to main memory, with is DDR3 1600.

(d) No other caches.

3. A complex 4 core system:

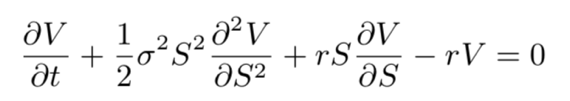
(a) Each core is Out-Of-Order (OOO) at 1 GHZ.

(b) Each core has a L1 cache of size 128kb.

(c) Each core also has an L2 cache of size 1MB

(d) All processors are connected to a shared L3 cache of size 64MB.

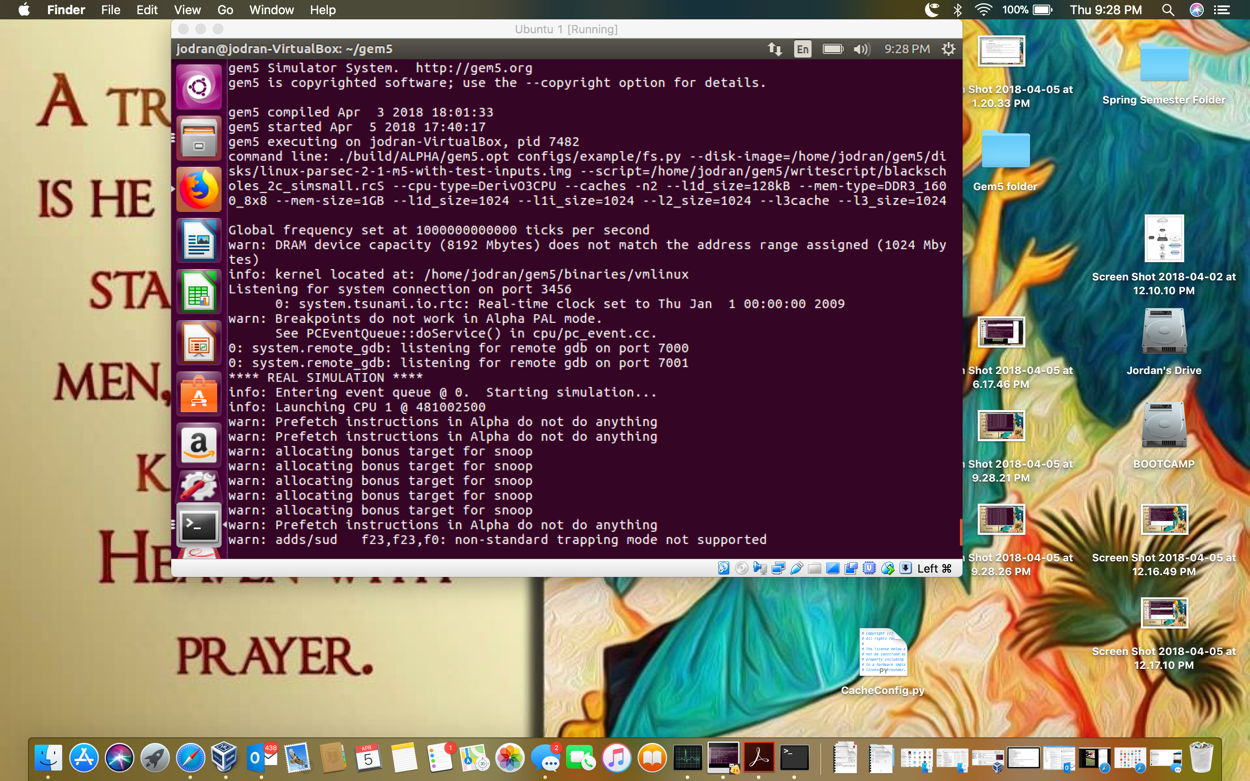
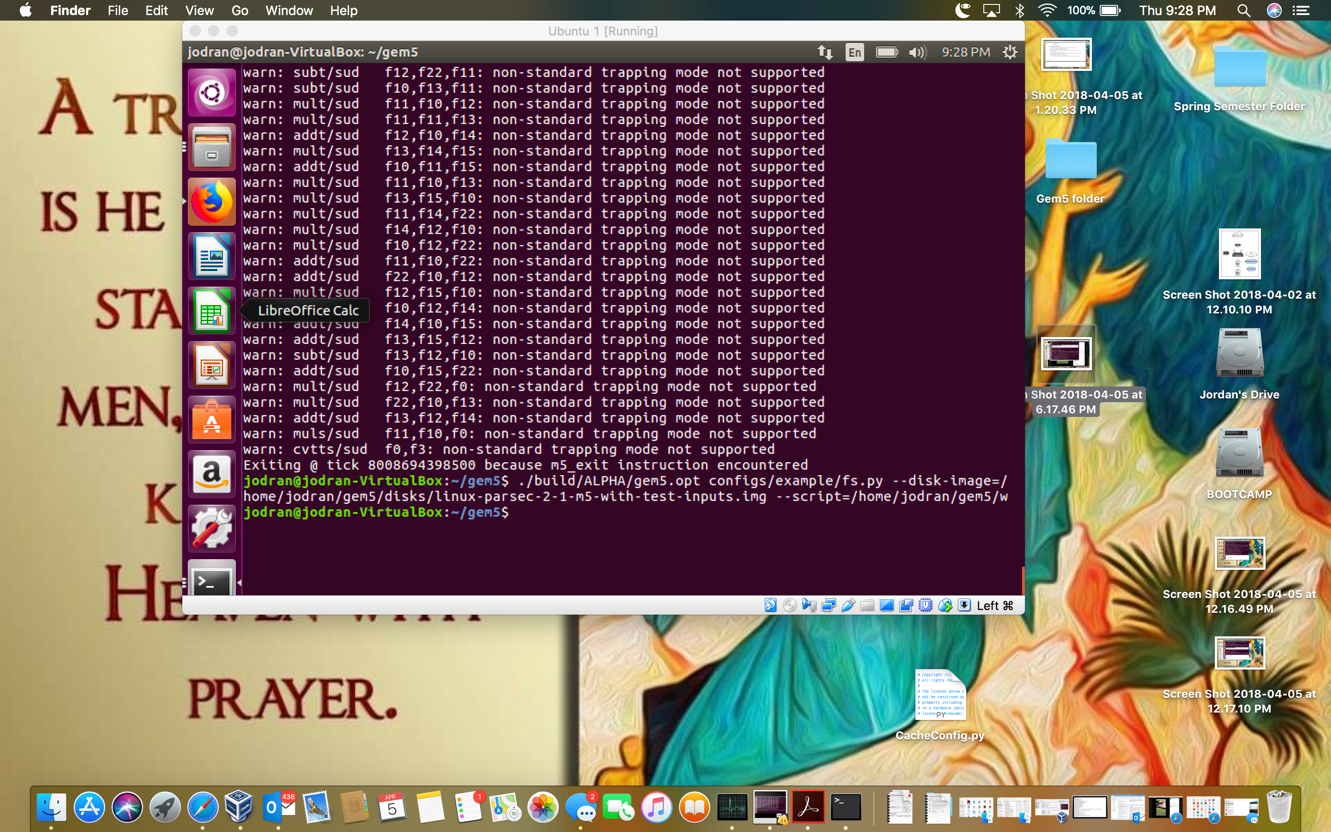
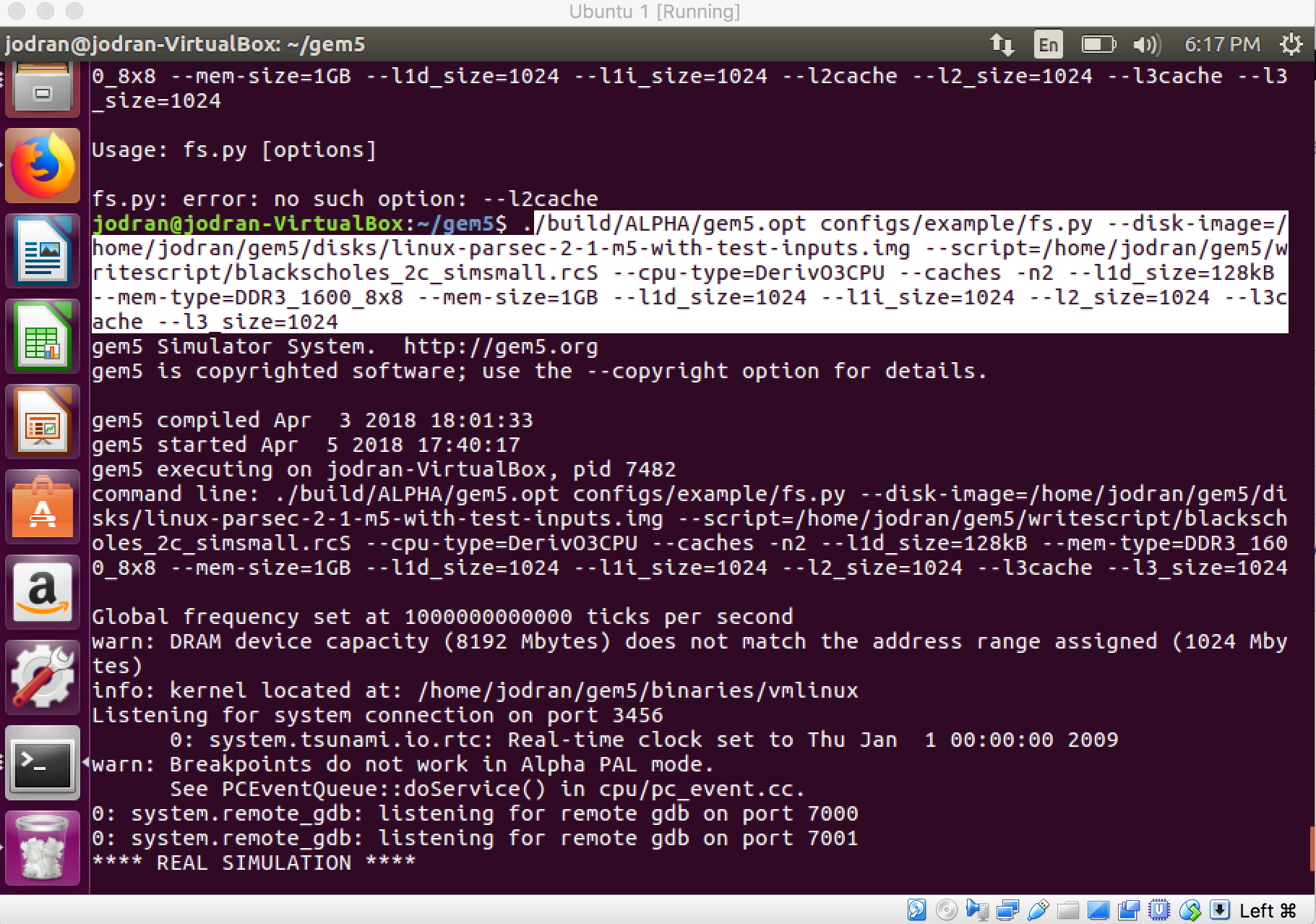
The benchmarks I chose to use were Blackscholes, Bodytrack and Canneal. These benchmarks are apart of a suite developed by Princeton university researchers. The use of Blackscholes, The blackscholes application is an Intel RMS benchmark. It calculates the prices for a portfolio of European options analytically with the Black-Scholes partial differential equation.



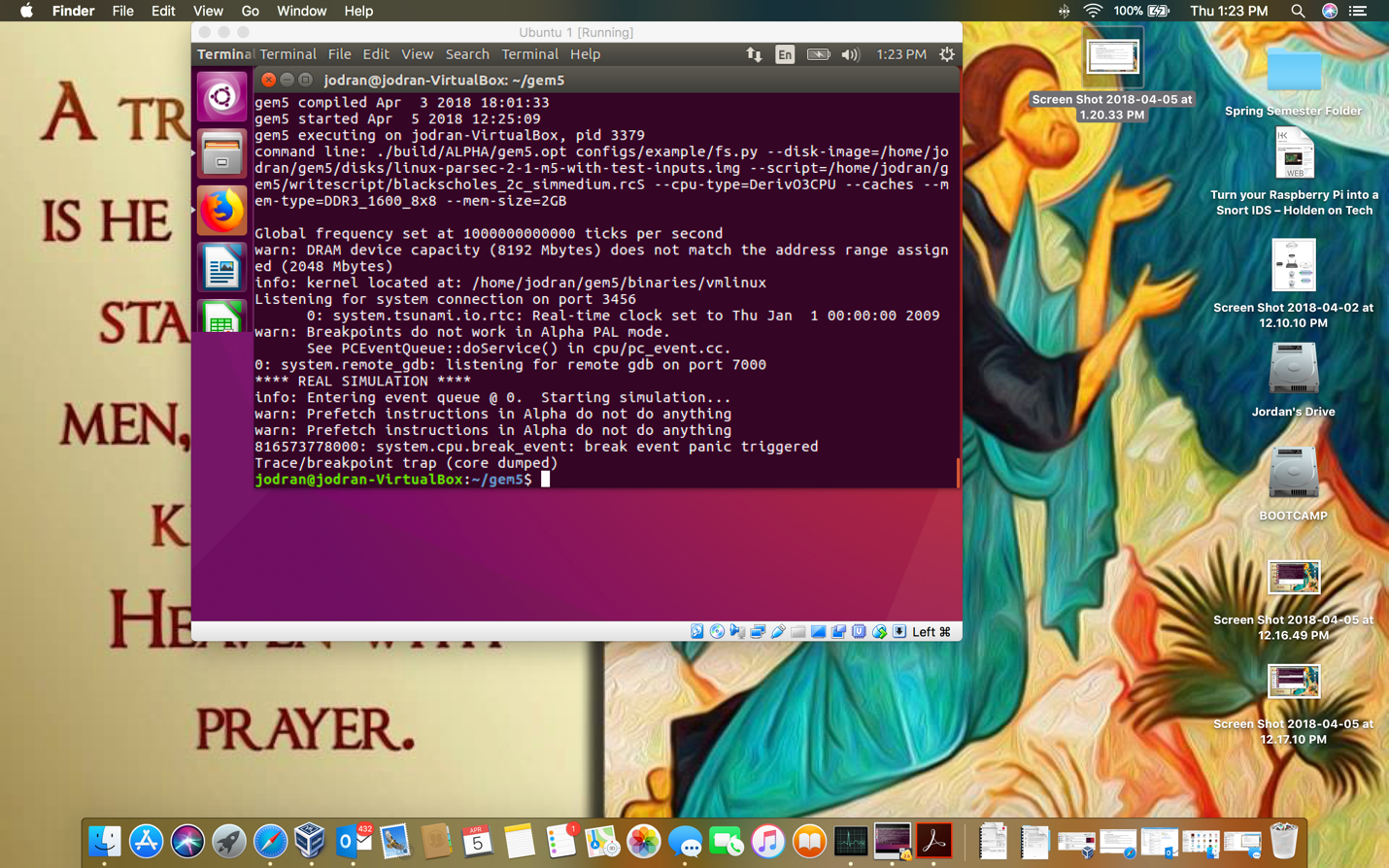
where V is an option on the underlying S with volatility σ at time t if the constant interest rate is r. There is no closed- form expression for the Black-Scholes equation and as such it must be computed numerically. The blackscholes benchmark was chosen to represent the wide field of analytic PDE solvers in general and their application in computational finance in particular. The program is limited by the amount of floating-point calculations a processor can perform.

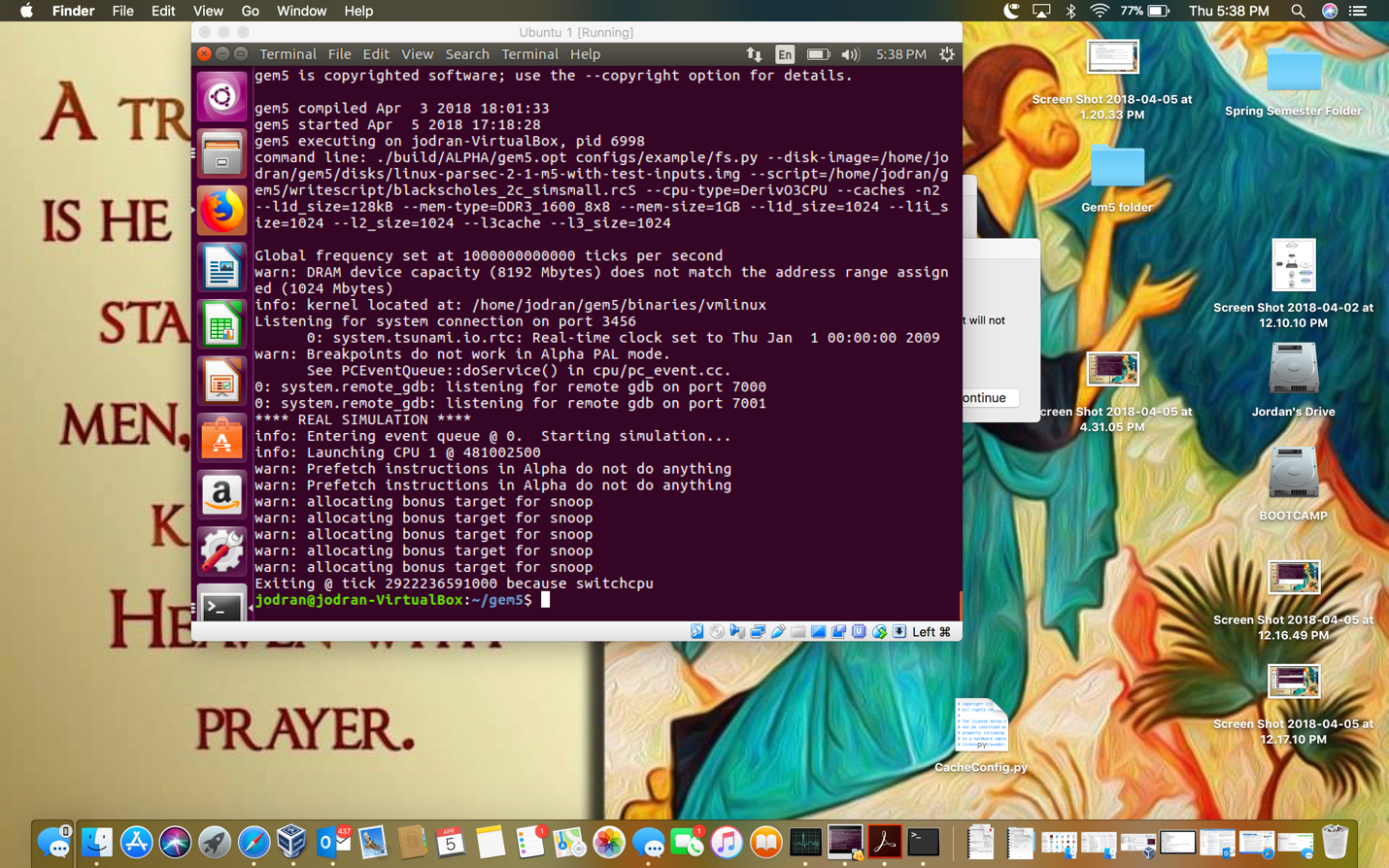
The bodytrack computer vision application is an Intel RMS workload which tracks a 3D pose of a marker-less human body with multiple cameras through an image sequence. Bodytrack employs an annealed particle filter to track the pose using edges and the foreground silhouette as image features, based on a 10 segment 3D kinematic tree body model. These two image features were chosen because they exhibit a high degree of invariance under a wide range of conditions and because they are easy to extract. An annealed particle filter was employed in order to be able to search high dimensional configuration spaces without having to rely on any assumptions of the tracked body such as the existence of markers or constrained movements. This benchmark was included due to the increasing significance of computer vision algorithms in areas such as video surveillance, character animation and computer interfaces.

The Canneal benchmark uses cache-aware simulated annealing (SA) to minimize the routing cost of a chip design. SA is a common method to approximate the global optimum in a large search space. Canneal pseudo randomly picks pairs of elements and tries to swap them. To increase data reuse, the algorithm discards only one element during each iteration which effectively reduces cache capacity misses. The SA method accepts swaps which increase the routing cost with a certain probability to make an escape from local optima possible. This probability continuously decreases during runtime to allow the design to converge. The program represents engineering workloads, for the fine-grained parallelism with its lock free synchronization techniques and due to its pseudo random worst case memory access pattern.

In my initial runs I was unable to successfully complete the proscribed architectures. Running each simulation with the given parameters I was posed with several errors, core dump, switchCPU, allocation errors. I was able to successfully run a the blackscholes benchmark with an L2 cache and a L3 cache shown in the picture below. 

I also attached the output file for this simulation. its names “SimOutput.txt”. the following errors were persistent when trying to run the first 2 architectures. See pictures below.





I’ve tried a number of different architecture to get a simulation to successfully run. Several different methods were done in order to get a successful simulation. Editing several different config files, editing the benchmark script, adding an L3 cache to the cache.py config file. None of my efforts were successful. I believe better documentation and guidelines would have benefited me in successfully completing the architectures and simulations.