Dynamically Changing CacheAssociativity and Size in L1/L2

**Cletus Menezes**

**G00948484**

**Final Project ECE 611**

1. **Project Description**

* In this project, I look to analyze the effects of dynamically changing the L2 cache and associativity size by analyzing the individual performance parameters.
* I have analyzed it based on two target functions (IPC/Power, IPC2/Power) during different phases of an application.
* The simulation involves different L1 and L2 cache configurations with 16 other different configurations. The target functions mentioned are then calculated based on the L1 and L2 cache power dissipation values that are obtained using CACTI.
* Modifications to the core code of SMTSIM is required so that it can periodically print the values of IPC for every 10K instructions.
* Based on this calculated IPC values and IPC/Power, IPC2/Power an estimated is made to determine the efficient configuration suitable for every particular application.

1. **Implementation**
   * The implantation of this projected needed some modification to the files of SMTSIM. These modifications are needed to periodically obtained the values of IPC for every 10k instructions of the commit stage
   * A conditional loop is needed to report he IPC values for every 10k instructions until the execution of 10 million instructions.
   * At the same time the committed instructions and the number of clock cycles also need to be recorded and printed.
   * Based on the total number of committed instructions and the number of cycles between two sampling intervals, the total IPC value is calculated for the specific interval.
   * These changes need to be made in the run.c file of SMTSIM
   * Also changes need to be made to the cache.c file of SMTSIM also needs to be to prevent it from printing the cache accesses and the ticks that are a default feature in the code.
   * The simulation was run for two of the two core benchmarks namely Applu~Bzip and Cactus~Apsi.
   * Before running the two\_core.py file SMTSIM needs to be built to incorporate the changes in the SMTSIM environment. The simulation is done based on the following table that shows the configurations used for the 16 simulations.
   * The output files are generated using the shell files that were previously generated. These files contain the IPC values along with the clock cycles and the total number of instruction executed instructions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Configuration No.** | **L1 I-cache, D-cache Size** | **L1 I-cache, D-cache Associativity** | **L2 cache size** | **L2 cache associativity** |
| 0 | 32768 | 1 | 131072 | 4 |
| 1 | 32768 | 1 | 131072 | 8 |
| 2 | 32768 | 1 | 262144 | 4 |
| 3 | 32768 | 1 | 262144 | 8 |
| 4 | 32768 | 2 | 131072 | 4 |
| 5 | 32768 | 2 | 131072 | 8 |
| 6 | 32768 | 2 | 262144 | 4 |
| 7 | 32768 | 2 | 262144 | 8 |
| 8 | 65536 | 1 | 131072 | 4 |
| 9 | 65536 | 1 | 131072 | 8 |
| 10 | 65536 | 1 | 262144 | 4 |
| 11 | 65536 | 1 | 262144 | 8 |
| 12 | 65536 | 2 | 131072 | 4 |
| 13 | 65536 | 2 | 131072 | 8 |
| 14 | 65536 | 2 | 262144 | 4 |
| 15 | 65536 | 2 | 262144 | 8 |

Table for the 16 different configurations

* Using another python files called get.py I have extracted the IPC values from each of the out files and exported them to an excel sheet in the .csv format.
* The values for IPC/Power and IPC2/Power were obtained online from CACTI based on the input configurations used for each simulation
* A table for all the 16 configs for IPC/Power and IPC2/Power along with IPC is created for both the benchmarks
* After calculating all the values, the highest value for the IPC/Power and IPC^2/Power are calculated and these values are used to plot the graphs showing the performance of the 16 different configurations.

1. **Graphs**

* **Applu- Bzip2\_source**

From the graphs above it can be easily seen that the configuration 0 has the best performance parameters in terms of IPC/Power and IPC2/Power among all the 16 configurations. However, the target values for configuration 8 is also very close to that of the performance of config 0.

* **Crafty-Apsi**

From the graphs above it can be seen that configuration 0 and 8 have the best target values in terms of IPC/Power and IPC2/Power for Crafty~Apsi. Both of these have equal values.

A table summarizing all these target values for both the benchmarks have been shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Applu~Bzip2\_source2** | | **Cactus~Apsi:** | |
| **Configuration No.** | **Highest IPC/Power** | **Highest IPC^2/Power** | **Highest IPC/Power** | **Highest IPC^2/Power** |
| 0 | 1.074402239 | 0.752435046 | 2.467802664 | 3.969685155 |
| 1 | 0.456732311 | 0.316067894 | 1.061667993 | 1.707789578 |
| 2 | 0.963280666 | 0.666748199 | 2.238666861 | 3.601099365 |
| 3 | 0.459047771 | 0.321461973 | 1.054464091 | 1.696201446 |
| 4 | 0.883729351 | 0.618814687 | 2.030128743 | 3.265646826 |
| 5 | 0.420892346 | 0.293304724 | 0.971557779 | 1.562839099 |
| 6 | 0.815554897 | 0.571397334 | 1.872465499 | 3.01203115 |
| 7 | 0.42047473 | 0.29455306 | 0.965521382 | 1.553129006 |
| 8 | 1.060806696 | 0.733512824 | 2.467802664 | 3.969685155 |
| 9 | 0.459737285 | 0.32024058 | 1.061667993 | 1.707789578 |
| 10 | 0.975122592 | 0.683242072 | 2.238666861 | 3.601099365 |
| 11 | 0.45898353 | 0.321372006 | 1.054464091 | 1.696201446 |
| 12 | 0.884038554 | 0.61924779 | 2.030128743 | 3.265646826 |
| 13 | 0.420950932 | 0.293386383 | 0.971557779 | 1.562839099 |
| 14 | 0.806320562 | 0.558530996 | 1.872465499 | 3.01203115 |
| 15 | 0.415772544 | 0.288001899 | 0.965521382 | 1.553129006 |
| **Adaptive** | **1.074402239** | **0.752435046** | **2.467802664** | **3.969685155** |

1. **Conclusion:**

* From my observations, it can be seen that even for the most efficient configuration among the 16 different configurations the value of IPC/Power and IPC^2/Power goes down in some phases.  The dynamic model can and will improve the overall performance by adapting the configuration parameters to bring back the performance indexes back to the higher level.
* By measuring IPC/Power and IPC2/Power parameters and by evaluation of the same I can conclude that for the execution of 10 million instruction the model proposed for a dynamically changing cache based on L1 and L2 caches is efficient and will increase the ratio of performance to power consumption of the processor.
* The applications with changing locality of reference are the ones which will benefit the most from this model. However, applications that don’t exhibit a changing locality reference may not benefit from this model.
* The reason is that the processing overhead that is inured as a result of the dynamically changing cache parameters system will not be feasible for such applications that would not be able to benefit from the proposed model. This could in return have a negative effect on applications that don’t have a varying locality of reference.