CS752 Project Proposal: Load Value Prediction

Prajyot Gupta, Satvik Maurya, Robert Viramontes

Topic Description

In our project, we intend to explore the concept of value locality and load value prediction, as proposed in [1]. We will characterize the performance of a load value prediction scheme on modern workloads and enhance the prediction scheme by using a combination of predictors (possibly including a perceptron-based predictor).

Phase 1

In Phase 1, we will be running SPEC CPU 2006 Benchmark, based on 10 workloads, to get an estimate of simulation runtimes and computational power required for project implementation based on checkpoints availed for these workloads. We will also be reviewing the O3 CPU code to understand where our load value prediction objects will need to wire in to modify the CPU behavior.

Phase 2

In Phase 2, we will be re-implementing the load value prediction design proposed in [1] and comparing the results on modern SPECCPU workloads. We will implement the primary functional units of the design within the existing gem5 CPU model, including the load value prediction table (LVPT) and the Load classification table (LCT), Constant Verification Unit (CVU).

- LVPT – Takes in the instruction and predicts the data. Contains history of last ‘n’ unique values from given instruction.
- LCT – This is a 2-bit saturation confidence counter which introduces hysteresis to the prediction.
- CVU – This unit verifies the speculated data with the actual data fetched from memory. It also generates the control signals for flushing instructions from the pipeline in case of a misprediction.

We will also modify the existing pipeline structures as necessary to interface with these new functional units. In particular, the LVPT and LCT will modify decode in the frontend of the pipeline and the CVU will be implemented in the backend before an instruction retires. The implementation of the CVU will need a recovery mechanism to be implemented as well to handle a misprediction. We will conclude Phase 2 by evaluating the design against a baseline CPU model without load value prediction to determine the impact of this modification.

Phase 3

In Phase 3, we plan to investigate improvements to the load value prediction algorithm and compare results of our evaluation metrics with the implementation from [1]. Our modifications to the value prediction algorithm will be based on our observations from Phase 1, but we are currently considering modifications informed by a perceptron-based branch predictor.

In short, our contributions will be 1) *evaluating the load value predictor design of* [1] *on modern workloads* and 2) *modifying the prediction algorithm using approaches similar to* [2] *&* [3] *to improve the efficacy of load value prediction in modern workloads*.

Proposed Timeline

Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**
Proposal			Progress Report		Status email	Status email	Presentation
Phase 1	Phase 1						Buffer
	Phase 2	Phase 2	Phase 2	Phase 2			Buffer
				Phase 3	Phase 3	Phase 3	Buffer

Motivation

As we have seen in lecture, in recent decades compute performance has increased at a faster rate than memory performance. This trend is commonly referred to as the memory wall. We are interested in investigating techniques that help hide the gap between compute and memory performance. Of the techniques we have studied and found in the literature, we are particularly interested in exploring the concept of value locality, as discussed in [1] and [4]. With value locality, the goal is to alleviate dataflow limits that result from true dependencies and expose more opportunity for instruction-level parallelism in serial programs. [1] was published in 1996 based on models of the PowerPC 620 and Alpha AXP 21164. We believe there is value in studying how this technique performs today in a recent gem5 model of an x86 processor with more modern workloads. When the speculative values are correctly predicted, we can hide the memory latency that would typically be incurred by true data dependencies.

Evaluation Methods

Tool – Gem5 simulator with x86 O3 CPU. The simulator will be run on the CS Department Linux machines and, if appropriate, batch jobs for benchmark runs may be submitted via Condor. We will use of the following simulator configurations:
L1 D$ & I$ cache: Size = 32kB, Associativity = 8; L2 cache: Size = 256kB, Associativity = 8;
L3 cache: Size = 8MB, Associativity = 16; Main Memory: Type = DDR3_2133_8x8, Size=16GB

Workloads – We plan to use the SPEC CPU 2006 workload to benchmark the performance as we have access to gem5 checkpoints that will simplify initial simulation efforts. We will use following ten subsets of workloads from the benchmark: astar, bzip2, gobmk, h264ref, hmmer, mcf, calculix, gromacs, povray and lbm. Checkpoints were availed for the workloads, with the above-mentioned configurations of for L1, L2, L3 caches and main memory to warm up all structures (Caches, branch predictor and value predictor) for 10B instructions. These workloads were first warmed up with 10 billion instructions and then simulated for the next 10 million. Time permitting, we would like to extend our workloads to include recent benchmarks like SPEC CPU 2017.

Reference Implementation – We will compare our results with those reported by Lipasti *et al* [1].

Metrics – Our primary metric will be the speedup of the benchmarks. We are also be interested in collecting metrics that profile the prediction performance, such as percent % of LVPT predictions correct. We will also compare the changes in the IPC, data cache accesses and number of squashed predictions.

References

[1] M. H. Lipasti, C. B. Wilkerson and J. P. Shen, "Value locality and load value prediction," *ACM SIGPLAN Notices*, vol. 31, p. 138–147, 9 1996.

[2] A. Perais and A. Seznec, "Practical data value speculation for future high-end processors," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, Orlando, 2014.

[3] R. Sheikh and D. Hower, "Efficient Load Value Prediction Using Multiple Predictors and Filters," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Washington, 2019.

[4] M. H. Lipasti and J. P. Shen, "Exceeding the dataflow limit via value prediction," in *Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture. MICRO 29*, Paris, 1996.

Progress Report – March 31, 2021

Phase 1- Completed

Benchmarks: We have selected the below mentioned workloads based on Seznec's, *"Practical data value speculation for future high-end processors", HPCA 2014* & *"Value Speculation through Equality Prediction", ICCD 2019*, wherein they have done a comparative analysis on the IPC values for LVP implementation for SPEC 2006 workloads. We have decided to use 6 INT workloads in `astar, bzip2, gobmk, h264ref, hmmer & mcf`; and 4 FP workloads in `calculix, gromacs, povray` and `lbm`. In the given workloads `bzip2`, `povray` and `h264` display high value locality as they use image compression, image processing and video compression algorithms, with locality inherent in their inputs.

Tabulated below are the parameters under consideration, which will be compared with the stats generated using LVP:

Type	SPEC Workloads	IPC Values	L1 D$ Miss Rate	L1 I$ Miss Rate	L2 $ Miss Rate
INT	`astar`	0.37406	0.050858	0.000028	0.156499
INT	`bzip2`	0.343934	0.037353	0.000028	0.577443
FP	`calculix`	0.68632	0.138516	0.000298	0.083284
INT	`gobmk`	0.567421	0.016133	0.027441	0.224647
FP	`gromacs`	0.895534	0.120091	0.00014	0.082671
INT	`h264ref`	1.112334	0.058384	0.004315	0.527771
INT	`hmmer`	0.874699	0.011225	0.000029	0.601825
FP	`lbm`	0.151453	0.302318	0.000043	0.607628
INT	`mcf`	0.421346	0.031346	0.000015	0.965158
FP	`povray`	1.382193	0.035847	0.008685	0.010826

Phase 2

- Created GitHub repository to store code and collaborate.
- Wrote documentation to outline the architecture of the software to implement the load value prediction, noting the major functional blocks and their required inputs/outputs.
- Wrote skeleton code for the major functional units to be implemented. Integrated skeleton code into the gem5 build process, including load value prediction debug flags.
- Completed the initial implementation of the load classification unit, which heavily based on the built-in 2bit_local branch predictor design. This returns an enumeration value depending on if the load at the instruction address is Unpredictable, Predictable or Constant.
- Modified the DerivO3 CPU implementation to query the load classification unit and load value prediction table when fetching a PC and confirmed via debug print statements that the LCT & LVPT are providing values.
- Implemented the CAM search APIs for the CVU- these will search the CAM in the CVU for either a `{LVPT index, data address}` pair or for a data address of a store instruction that is about to commit.
    - o Also identified areas of the Gem5 code which will incorporate these APIs- the LCT will pass the `{LVPT index, data address}` pair to the CVU when it predicts a load instruction as a constant and a store that reaches writeback (in `iew_impl.hh`) will pass its address to the CVU for comparison using the `processStoreAddress()` API.