Contents

Slide 1: Hi, welcome to the presentation on applying deep learning to prefetching

Slide 2: CPU's are designed so that they reduce latency as much as possible However, memory remains to be a significant bottleneck. So much so, that most modern applications spend about 50% of total cycles are spent waiting for data from memory.

Slide 3: Prefetching is the mechanism that loads data from an address in memory before it is needed, so that when it is needed by an instruction, it is already in the cache. This means that it can reduce the bottlenecks caused by memory operations We can improve a prefetcher by using an RNN to predict addresses

Slide 4: One particular method that a prefetcher loads an address is by looking at the stride, or delta, of recent memory addresses. For example, accesses to addresses 0, 4, 8, 12,... seem to be following a delta of 4, and so the prefetcher would load in addresses of multiples of 4 + offset. A different type of prefetcher is a correlation prefetcher. This tries to learn patterns in memory address accesses that aren't linear. These prefetchers are require large tables to store address access information which reduce performance and so they aren't used in modern multicore processors

Slide 5: Typically, memory address accesses are few and far between, this means that, when trying to predict an address, there are a large amount that should and often aren't accessed, having these as possibilities however, makes predicting addresses difficult. Neural Nets work best when data is normalised, however, if addresses are normalised, then a significant amount of data is lost. We can treat the problem of predicting an address as classifying the next word in a sequence. This means that the address space would be a vocabulary.

Slide 6: When executing a program, the addresses that the program uses may change with every execution. This can be caused due to factors such as Address Space Layout Randomisation. However, the behaviour of the execution doesn't change, and so, it is possible to learn the deltas for a program whilst not being able to predict the raw address itself. As a result, deltas are used as input rather than raw addresses

Slide 7: The embedded LSTM model is where each instruction that accesses memory has some data about it concatenated with the corresponding delta of that instruction. This then gets passed in as input to the LSTM The alternative model is to cluster instructions based on memory address locations, and then generating in-cluster deltas to pass in as input to the

LSTM

Slide 8: The embedded LSTM only models the most frequently occurring deltas, this helps keep the memory footprint low. The input deltas that are used are ones that occur 10 or more times in the dataset. The top 10 predictions made by this LSTM at this timestamp are loaded into cache. This model improves by having a larger vocabulary to work with, however this then incurs the fee of having a larger memory footprint

Slide 9: Clustering + LSTM work on the fact that data structures arrays and structs are stored in contiguous blocks of data and accessed repeatedly This model performs prefetching in a local context by using the deltas of instructions which access similar addresses. The clustering is done by K-Means and the Deltas are computed within each cluster. This model implements the Adagrad Algorithm

Slide 10: Adagrad is an adaptive gradient descent algorithm that has a per-parameter learning rate, which means that at every timestep, the weights for every parameter gets updated individually. The adaptation that Adagrad implements is that parameters that occur frequently have their learning rates reduced, whereas less frequently occurring parameters have a higher learning rate.

Slide 11: At each timestep, 10 predictions are made. If any of the deltas are correct, then that set of predictions is deemed to be correct. If the actual delta is outside of the model's vocabulary, then the set is an instant failure. The figure shows the performance of the embedded and clustering LSTM models against 2 other prefetchers. From this we can see that the 2 prefetchers that use the LSTM perform at least as good as the other prefetchers.

Slide 12: Thanks