Implementation and Optimization of Data Prefetching Algorithm Based on LLVM Compilation System

Group 4: Yixin Zhang, Jiazhen Zhao, Yuntao Zhang, Yuqing Wang

Yunda Chai et al 2021 J. Phys.: Conf. Ser. 1827 012136
Memory access optimization:
- Why we need memory access optimization?
  - Loading data from memory takes a lot of time.
- Current popular approach: principle of data locality (When loading a data from memory, load its neighbor data as well as a whole block)
  - Drawback: Poor data locality issue (which results in 64% CPU cycles loss)\([1]\), high cost of doing profiling to fix locality issue \([2]\).

Data prefetching
- What is prefetching in memory access optimization?
  - Prefetching is to read the data from memory into the cache before the data is used.
    (Eg, load instruction is executed in advance)

---

\[1\]: Tanvir Ahmed Khan, DMon: Efficient Detection and Correction of Data Locality Problems Using Selective Profiling, 15th {USENIX} Symposium on Operating Systems Design and Implementation ({{OSDI}} 21), Pages 163-181
\[2\]: Aleksey Pesterev, Nickolai Zeldovich, and Robert T Morris. Locating cache performance bottlenecks using data profiling. In Proceedings of the 5th European conference on Computer systems, pages 335–348. ACM, 2010
Without **data prefetching:**

With **data prefetching:** (loading process can be done in parallel with the program execution)
Benefits of **data prefetching**
- Parallelize the loading process with the rest of the program
- Reduce memory access latency to improve program performance
- Solve the issues related to data locality

Drawbacks:
- Additional costs to do the analysis of how to do the prefetching

Now, let’s show the algorithm for **data prefetching** ......
Algorithm Overview

➢ What:
  ● Iterate through all load/store instructions. Determine the capability and efficiency to prefetch the instruction.

➢ How:
  ● Traverse the CFG graph using depth-first algorithm.
  ● Iterate through qualified BBs. Each load/store will go through cost check.
Pre-condition Check

Two basic assumption:

- No prefetching in BB
- Objects that need prefetching exist

Parameter explanation:

- PrefetchDistance: determine whether to analyze
- InstrinsinID = prefetch: the instruction is prefetching
Sub-Algorithm: Prefetch Scheduling

➢ Purpose:
  • Schedule ahead to eliminate the overhead of inserting prefetch instructions.
  • Hide the delays of memory access. (elaborated in the introduction).

➢ Details:
  • Let’s define:
    • Tpref = Delay in number of clocks of issuing the prefetch instructions for a certain basic block.
    • Tloop = Number of clocks for one loop when there is no prefetch operation.
    • IterAhead = Number of loop iterations ahead to issue the prefetch instructions.
  • Then:
    • IterAhead = (Tpref + Tloop - 1) / Tloop
    • - 1 here, as we want to let the prefetch happens at least one cpu clock before the data access.
➢ Algorithm for prefetch scheduling

Algorithm 2. Basic block instruction cost and Prefetch scheduling distance

```plaintext
For each BB in CFG
    For each ins in BB
        if ins is ephemeral value
            continue
        LoopSize += ins’Cost
    end for
    if LoopSize is null
        LoopSize = 1
        ItearsAhead = (PrefetchDistance + LoopSize - 1) / LoopSize
    end for
```

* Ephemeral value means just a simple value. If an instruction is an simple value, we will skip’s its cpu clocks count.
Cost Model

- Do not prefetch if constant value or access step is not large enough

| Algorithm 3. The analysis of cost model |
|----------------------------------------|
| **For** each *ins* in loop            |
| */ Get the prefetch analysis information of the loop body*/ |
| if(The value of prefetch is an invariant || The step of access is not enough to guarantee prefetch || The prefetch operation overlaps) |
|   **continue**                         |
|   if(The prefetch scheduling distance > prefetch forward iteration value) |
|     **continue**                       |
| **end for**                            |
Results & Discussion

- LLVM, Shenwei platform, SPEC2006 test set
- average speedup of 6%

![Speedup Ratio Chart]
Conclusion

- Prefetch analysis algorithm
  - Loop analysis, scheduling, cost model
  - eliminate unnecessary prefetch operations
  - reduce instruction overhead
  - improve system performance

- Future improvement
  - Support outer-loop prefetch