Cache Behavior Modeling of Codes with Data-Dependent Conditionals

Authors :
Diego Andrade
Basilio B. Fraguela
Ramón Doallo

Computer Architecture Group
Departamento de Electrónica e Sistemas
Universidade da Coruña

Cache Behavior Modeling of Codes with Data-Dependent Conditionals

---

Introduction (I)

- The growing difference between the processor and the main memory speeds : bottleneck for systems performance.

- Optimal usage of the memory hierarchy : very important in embedded systems.

- Approaches to study cache behavior :
  - Trace-driven simulations.
  - Built-in hardware counters.
  - Modeling.

Introduction (and II)

- Systematic modeling strategy that allows a fast automated analysis that provides good levels of accuracy [FDZ03].

- Extension that enables the model to analyze codes with data-dependent conditionals whose condition follows an uniform distribution.

Basic model concepts (I)

- Support : K-way associative caches with LRU replacement.

- Kinds of misses :
  - Cold or compulsory miss.
  - Interference miss.

- Essential idea :

  ' In a K-way set with LRU replacement policy, a given line is replaced when K or more different lines mapped to its same cache set have been referenced since its last access '

Area Vectors

- Given a data structure V,

$$S_V = S_{V_0}, S_{V_1}, \ldots, S_{V_K}$$

where S_{V_0} it is the ratio of sets that have received k or more lines and S_{V_i}, $0 < i \leq K$ is the ratio of sets that have received k-i lines from the structure.

- The area vector associated to a given structure is calculated as a function of the access pattern.

Area Vectors Addition

- Algoritmo de suma de vectores de área :

$$(S_U \cup S_V)_0 = \sum_{j=0}^{K} (S_{U_j} \sum_{i=0}^{K-j} S_{V_i})$$

$$(S_U \cup S_V)_i = \sum_{j=i}^{K} S_{U_j} S_{V_{(k+i-j)}} \quad 0 < i \leq K$$

- It is based in the addition of independent probabilities. It does not take account the relative positions of data structures.

Initial Modeling Scope

```
DO $I_Z$=1, $N_Z$, $L_Z$
   ...
   DO $I_1$=1, $N_1$, $L_1$
     DO $I_0$=1, $N_0$, $L_0$
       A($f_{A1}$($I_{A1}$), $f_{A2}$($I_{A2}$), ..., $f_{AdA}$($I_{AdA}$))
       ...
       IF B($f_{B1}$($I_{B1}$), $f_{B2}$($I_{B2}$), ..., $fB_{dB}$($I_{BdB}$))
        C($f_{C1}$($I_{C1}$), $f_{C2}$($I_{C2}$), ..., $f_{CdC}$($I_{CdC}$))
       ...
     END DO
     ...
   END DO
END DO
```

Probabilistic Miss Equations (I)

- Our method generates a Probabilistic Miss Equation (*PME)* for each reference in each nesting level.

- Loops are examined beginning in the innermost loop containing the reference and proceeds outwards.

- The PME for a given reference and nesting level is built recursively using the PME for that reference in the inmediately inner loop.

Probabilistic Miss Equations (and II)

- In each nesting level, the equation depends on the area vector associated to the regions accessed since the last reference to a given line of the data structure referenced by R.

- If the reference is guarded by a conditional sentence, we need to know the probability *p* that the condition be true.

Condition Independent Reference Formula

$$F_i(R, S(\text{RegInput}), p) = L_{R_i} F_{i+1}(R, \text{RegInput}, p) +$$
$$(N_i - L_{R_i}) F_{i+1}(R, S(\text{Re}\, g(A, i, 1)), p)$$

$$L_{R_i} = 1 + \left\lfloor \frac{N_i - 1}{\max\{L_s / S_{R_i}, 1\}} \right\rfloor$$

Condition Dependent Reference Formula

$$F_i(R, S(\text{RegInput}), p) =$$
$$L_{R_i} TPR(p_i, S(\text{RegInput}), i, G_{R_i}, p)$$

$$TPR(p_i, S(\text{RegInput}), i, n, p) =$$
$$\sum_{j=1}^{n} PR(p_i, S(\text{RegInput}), i, j, p)$$

Validation (I)

```
DO I = 1,M
     X=A(I)
     DO J=1,N
          Y=B(J)
          IF (B(J).GT.K) THEN
          C(J) = X+Y
          ENDIF
     ENDDO
 ENDDO
```

Synthetic Kernel

Validation (II)

M	**N**	**p**	**Cs**	**Ls**	**K**	**ΔMR**	**ΔNM**	**T.Sim.**	**T.Exec.**	**T.Mod.**
50000	47500	0.2	65536	16	2	5.067	0.372	141	60	0.005
22000	14500	0.4	16384	8	4	1.260	0.239	21	6	0.005
18000	22000	0.1	4096	32	4	0.417	0.141	95	8	0.004
17500	17500	0.3	8192	8	4	0.341	0.063	22	7	0.004

Validation (III)

```
DO I=1,M
      DO J=1,P
            T=0
            DO K=1,N
                  IF (A(I,K).NEQ.0) THEN
                        T=T+A(I,K)*B(K,J)
                  ENDIF
            ENDDO
             C(I,J)=C(I,J)+T
      ENDDO
ENDDO
```

Matrix Product

Validation (and IV)

M	**N**	**P**	**p**	**Cs**	**Ls**	**K**	**ΔMR**	**ΔNM**	**T.Sim.**	**T.Exec.**	**T.Mod.**
1700	1600	1250	0.2	8192	8	4	0.018	0.017	342	162	0.023
750	750	1000	0.4	32768	4	2	0.260	0.040	56	31	0.019
900	850	900	0.1	65536	8	1	0.525	0.065	48	29	0.020
1000	850	900	0.3	4096	8	1	0.074	0.054	70	40	0.013

Conclusions and future work (I)

- Analytical modeling of codes with regular access patterns.
- We have extended the model to analyze codes with data dependent conditionals.
- The method is based in obtaining the PMEs.

Conclusions and future work (II)

- The model can be applied in an automated fashion: integrated in Polaris Compiler.
- It is a very valuable tool to guide the optimization process of a compiler
  - Fast
  - Accurate

Conclusions and future work (and III)

- We are currently developing tools to guide transformations of the code using our model.

- Future lines of research include the extension of the model to consider :
  - Non-uniform distributions.
  - Codes with irregular access patterns due to the use of indirections.