Weekly Report

Weiping Liao

01/29/2005

1 Performance Model

The performance model is a M/D/1/-/N queue, where N is the number of PEs. The feature of this queue is the finite source since we have finite number of PEs. The formula for this kind of queue is provided in [1] and lists as follows, When there are k requests in the queue (pending requests from PEs waiting for main memory access), the arrival rate $\lambda_k = \lambda_0 * (N - k)$, where 0<=k<=N. λ_0 is the memory request rate for a single PE, and equal to $\frac{M}{C_p}$, where M is the number of memory requests and C_p is the cycles consumed by pipeline structure. A mistake we made before is set λ_0 to $\frac{M}{C_p+M*T_a}$ where T_a is the average memory latency. That was wrong according to the queueing theory.

Suppose the memory can handle request in the rate of R_m, then the utilization rate $r = \frac{lambda_0}{R_m}$. Suppose p_i represent the probability of where i pending requests in the system (including memory unit and waiting queue), then p_i is given by the following:

$$p_i = \frac{N!}{(N-i)!} * r^i * p_0 \quad (1)$$

$$p_0 = \frac{1}{\sum_{k=0}^{N} k * r^k * \frac{N!}{(N-k)!}} \quad (2)$$

Given p_i, the total number of requests L can be calculated as $\sum_{k=1}^{N} k * p_k$. And the total number of requests currently waiting in the queue is $L_q = L - r(N-L)$. Further from Little's formula we can get the average waiting time $T_w = \frac{L_q}{\lambda_0(N-L)}$

The new model achieves smaller error (up to 8%) compared to that in our DAC submission (10%). Essentially we use a simple M/D/1 queue in the DAC submission. More importantly, the new model keeps the similar error bound for different memory bandwidth (I have tested 1/16, 1/32 and 1/64 word/cycle.), while our previously can not hold 10% error bound when the memory bandwidth decreases to less than 1/32 word/cycle.

2 Multi-core Optimization

Here we refine the problem. The goal is the same: for given die area, design multi-core microprocessor to maximize performance in terms of total throughput (TP).

Assuming we already have a single core microarchitectural configuration, denoted as S_F, and the given benchmark B, then we can obtain the following:

- The number of cores $N(S_F)$, which a function of S_F. This value is calculated by $total\_area\_for\_cores/area\_of\_single\_core$, where $area\_of\_single\_core$ is fixed once the S_F is decided.
- Throughput of single core $TP_s(S_F, B)$.
- L2 cache miss rate $MR_s(S_F, B)$.

The total throughput of the multi-core processor is given as $TP_{all} = N(S_F) * TP_s(S_F, B) - f_b(N(S_F), MR_s(S_F, B))$, where f_b is the function of bus overhead estimated by the previous section.

The missing parts in the above description include: (1) for single-core configuration optimization, so far we still focus on CPI. We should find a way to explore optimal throughput, and (2) the bus performance model should be able to handle different access rates. Once we fixed the two parts, the whole problem can be solved by TPWL easily.

The results we get can be compared to two cases: (1) idea case: no branch misprediction, no cache misses, no bus contentions; and (2) existing architectures: simply combine a few existing processors (Xeon or Opteron) together on the chip.

3 Journal revision

I rewrote the introduction, but am still working on the cache decay method.

References

[1] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*. John Wiley & Sons, Inc, 1998.