Feedback-Driven Threading: Power-Efficient and High-Performance Execution of Multi-threaded Workloads on CMPs.

Sondre Lefsaker - Runar Olsen
Introduction

- Processors today are headed towards more and simpler cores.
- Multi-threading for ease of programming vs performance
- Number of threads often locked to number of cores
- Increasing number of threads might lower performance
Introduction

- Authors observe that off-chip bandwidth and data synchronization often become bottlenecks.
- FDT proposed to lower power usage and reduce execution time by dynamically selecting number of threads.
- Two models based on FDT suggested to mitigate these problems, SAT and BAT.
- Experiments run on simulated homogeneous 32-core x86 processor. 3-level cache with private L1 and L2.
SAT - Problem

GetPageHistogram(Page *P)

For each thread: {

  /* Parallel part of the function */
  UpdateLocalHistogram(Fraction of Page)

  /* Serial part of the function */
  Critical Section:
      Add local histogram to global histogram
      Barrier

  }

Return global histogram
SAT - Model

\[ T_P = \frac{T_{NoCS}}{P} + P \cdot T_{CS} \]

\[ P_{CS} = \sqrt{\frac{T_{NoCS}}{T_{CS}}} \]

**Figure 6.** Example for analyzing impact of critical sections
SAT - Algorithm

Training loop executes a few iterations using one thread to measure Tcs and Tnocs. It sets TRAIN to false when training completes.

Estimate Pcs using Tcs and Tnocs.

Spawn Pcs threads to execute remaining iterations.

Figure 7. Synchronization-Aware Threading
**SAT - Results**

**Figure 8.** Performance of SAT. Vertical axis shows the normalized execution time and horizontal axis shows the number of threads.

**Figure 10.** Performance of SAT for 2.5KB and 10KB page-size.
**BAT - Problem**

EuclideanDistance(Point A)

/*Parallel threads compute partial sums*/
for i = 1 to N
    sum = sum + A[i] * A[i]

Return sqrt(sum)

**Figure 3.** A function which computes Euclidean Distance (ED) of a point from origin in an N-dimensional space

**Figure 4.** (a) Normalized execution time with respect to one thread for ED. (b) Bandwidth utilization of ED
BAT - Model and Algorithm

Figure 11. Example for analyzing bandwidth limited systems

\[ BU_P = P \cdot BU_1 \]

\[ P_{BW} = \frac{100}{BU_1} \]
**BAT - Results**

**Figure 12.** Performance of BAT. Vertical axis denotes normalized execution time and horizontal axis the number of threads.

**Figure 13.** Performance of BAT as off-chip bandwidth is varied.
SAT+BAT

On average: 17% reduction of execution time.
59% reduction in power usage.
Figure 15. Execution time and power for (SAT+BAT) and an oracle scheme. The values are normalized to 32 threads.