Understanding Communication Characteristics in Shared Memory Applications

Berkeley Parlab

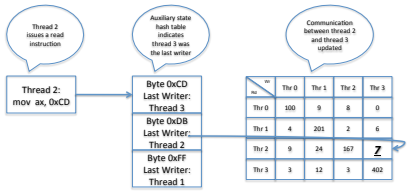
David Sheffield, Kurt Keutzer

Motivation

- Communication in shared memory programs is expensive
  - Avoid communication if possible
- We have developed a suite of tools to measure communication and memory behavior
  - Developer can visualize memory behavior
- Dynamic binary instrumentation with Pin
  - Instrument existing x86 binaries
    - No source modifications required
  - Analysis code is injected into an existing binary at runtime
    - Just-in-time recompilation

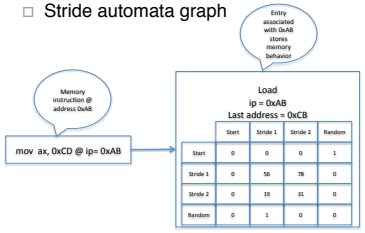
Measuring communication

- Metadata associated with main memory keeps track of sharers
- Memory operations augmented with analysis code
  - Dynamic binary translation used to inject instrumentation code



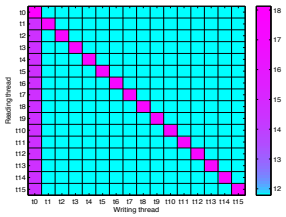
Recording memory behavior

- Each memory operation stores thread-local metadata
  - Last address
  - Stride automata graph



Splash-2 FFT

- 2^{20} complex points, 16 processors, six-step 1d FFT algorithm
- Results closely match our communication expectations
  - 2^{17} bytes exchanged between processors
    - Each processor owns $\frac{n}{p^2}$ complex doubles
    - Two all-to-all exchanges with 16 byte operands
      - First transpose is integrated into data distribution from processor 0



	Thread 0	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8	Thread 9	Thread 10	Thread 11	Thread 12	Thread 13	Thread 14	Thread 15
Thread 0	2245317	131078	131078	131072	131072	131072	131078	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 1	2245318	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 2	2245318	131078	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 3	2245317	131078	131078	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 4	2245317	131078	131078	131072	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 5	2245317	131078	131078	131072	131072	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 6	2245317	131078	131078	131072	131072	131072	2621440	131072	131072	131072	131072	131072	131072	131072	131072	131072
Thread 7	2245317	131078	131078	131072	131072	131072	131072	2621440	131072	131072	131072	131072	131072	131072	131072	131072
Thread 8	2245317	131078	131078	131072	131072	131072	131072	131072	2621440	131072	131072	131072	131072	131072	131072	131072
Thread 9	2245317	131078	131078	131072	131072	131072	131072	131072	131072	2621440	131072	131072	131072	131072	131072	131072
Thread 10	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	2621440	131072	131072	131072	131072	131072
Thread 11	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	131072	2621440	131072	131072	131072	131072
Thread 12	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	131072	131072	2621440	131072	131072	131072
Thread 13	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	2621440	131072	131072
Thread 14	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	2621440	131072
Thread 15	2245317	131078	131078	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	131072	2621440

CS267 programs

- Particle simulation
  - Shared memory implementation poorly scales due to excess communication

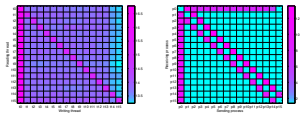


Figure 12: Pthread particle simulation with 16 threads Figure 13: MPI particle simulation with 16 threads

- 0-1 knapsack problem
  - UPC and shared memory implementation have similar communication topology

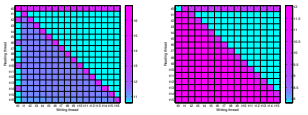
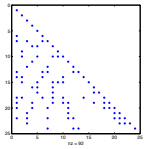
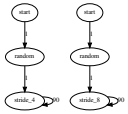


Figure 10: Pthread version 0-1 knapsack problem with 17 threads Figure 11: UPC version of 0-1 knapsack problem with 16 threads

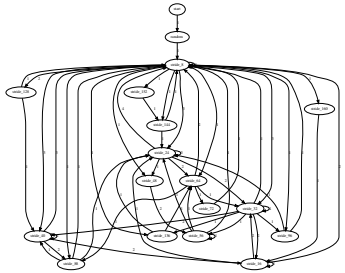
SpMV memory behavior



- Test matrix
  - CSR format
  - 92 non-zero entries
- Constant stride access to matrix and column indices
  - 4 byte operands used as column indices
  - 8 byte operands used for matrix values



- Access to the dense vector is highly irregular
  - Access pattern is data-dependent



Future work

- Apply tools to real applications
  - Splash-2 and CS267 assignments are not very realistic
- Reduce tool memory requirements
  - Redundant metadata
- Automatically detect computational patterns?
  - We can we cluster computational patterns with memory FSMs?