A Problem or an Opportunity?

Database workload

+ low throughput (0.8 IPC on an 8-wide superscalar)
+ naturally threaded application
+ widely used
- already high cache miss rates on a single-threaded machine

Key question:

Can SMT's simultaneous multi-thread instruction issue hide the latencies from additional misses in this already memory-intensive databases?

Database Bottom Line

SMT gets good speedups on commercial database workloads

- 3x for debit/credit systems, 1.7 for big search problems

The reasons:

- Limits inter-thread data interference in the caches to superscalar levels by:
  - a virtual-to-physical page mapping policy that exploits temporal locality in the physically-addressed L2 cache
  - address offsetting for thread-private data in the L1 data cache
- Exploits SMT's inter-thread instruction sharing (35% reduction in instruction cache miss rates)
- Hides latencies of all kinds with simultaneous, multi-thread instruction issue

SMT Performance