OpenMP

- Shared address space programming
- OpenMP is a “high-level” approach compared to Posix threads that is useful in many applications
- Comprised primarily of compiler directives (pragmas) as well as a library and environment variables
- Example: directive/clauses
  `#pragma omp parallel default(none)`
- Online reference:
  https://computing.llnl.gov/tutorials/openMP

Common OpenMP Directives

- #pragma omp parallel
  - Creates a group of threads for the following block of code
  - “top-level” directive; other directives are combined with this one
- #pragma omp for
  - Split a for loop among threads
  - Various “schedules” are available (most useful: static, dynamic)
- #pragma omp sections
  - Define a section for each thread
- #pragma omp critical
  - Define critical sections, where only one thread executes at a time (in sequence)
- #pragma omp single
  - Define a section of code where only one thread executes (others skip)

Common OpenMP Clauses

- parallel
  - if
  - num_threads
  - private/shared
  - reduction
- for
  - private/shared
  - reduction
  - nowait
- sections
  - private
  - reduction
  - nowait
- critical
  - There are no clauses for critical (but critical sections can be named)
- single
  - private
  - nowait

Compiling and Running OpenMP Programs

- `gcc –fopenmp progname.c`
- Set the number of threads (in increasing priority)
  - Use the environment variable `OMP_NUM_THREADS`
  - Use the function `omp_set_num_threads(num)`
  - Set the number of threads in the `parallel` pragma

OpenMP for vector addition

How much faster is using multiple threads compared to using a single thread?

Potential issues

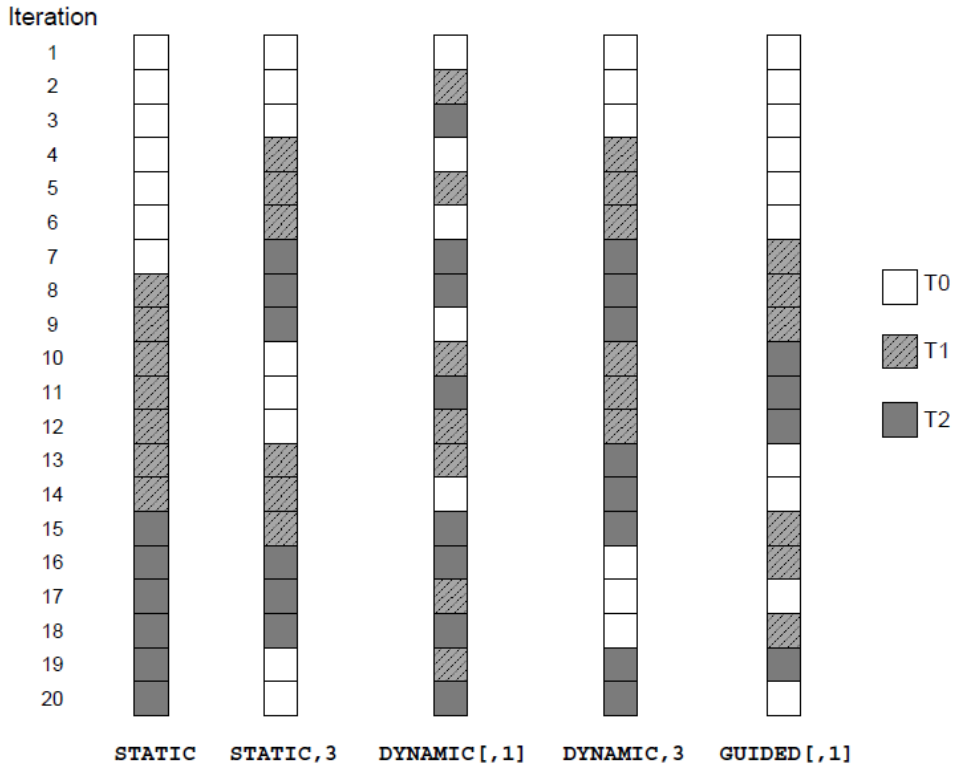
- Memory bandwidth limitations
- Thread creation overhead
- Thread scheduling overhead
- Conflicts in shared caches

Using multiple threads

- Memory bandwidth may limit speedup when using multiple threads
- One thread cannot fully access the memory bandwidth available
  - try the stream benchmark with multiple threads

OpenMP Thread Scheduling

- Static
  - Fixed partitioning of iterations
  - Least overhead
- Dynamic
  - Use if loop iterations have unequal cost
  - Thread gets a new “task” when it is available to work
- Guided
  - Chunk size tapers from large to small
  - Tries to avoid high overhead of dynamic scheduling



Thread Scheduling