Parallelisierung mit OpenMP

Lehrstuhl Seminar

Jonas Sternisko

Institut für Informatik
Albert-Ludwigs-Universität Freiburg

6. Mai 2013

Example: Execute a for-loop in parallel

Listing 1: Baseline

```
vector<TransferPatterns> patterns(_network.numStations());
for (size_t station = 0; station < _network.numStations(); ++station) {
  // Some costly operation, iterations are pairwise independent.
  TransferPatterns p = computeTransferPatternsFromStation(station);
  // Store the patterns.
  patterns[station] = p;
}
```

Listing 2: OpenMP parallelizes the execution at runtime.

```
vector<TransferPatterns> patterns(_network.numStations());
#pragma omp parallel for
for (size_t station = 0; station < _network.numStations(); ++station) {
  // Some costly operation, iterations are pairwise independent.
  TransferPatterns p =
    computeTransferPatternsFromStation(station);
  // Store the patterns, the vector has shared access.
  patterns[station] = p;
}
```

Parallelization using OpenMP

Open MP

- Preprocessor directives
  - Enabled by compiler flag `-fopenmp`
- Runtime library
  - `omp_get_max_threads()`
  - `omp_set_num_threads(n)`
  - `#include <omp.h>`
- Environment variables
  - `OMP_NUM_THREADS`, ...

Preprocessor Directives

Data Access

Listing 3: Default: Shared access

```
int value = 0;
#pragma omp parallel for shared(value)
for (size_t i = 0; i < N; ++i) {
  value += i;
}
```

- Write/Read access to `value` is synchronized

Listing 4: Private access

```
int value = 0;
#pragma omp parallel for private(value)
for (size_t i = 0; i < N; ++i) {
  value += i;
}
```

- Each thread gets its private copy of `value`.

Preprocessor Directives

Data Access

Listing 5: Final reduction operation

```
int value = 0;
#pragma [...] reduction(+:value)
for (size_t i = 0; i < N; ++i) {
  value += i;
}
```

Reduction

- Each thread gets its own copy of `value`
- Finally, values are combined using `operator+`
- Supports any binary operator `+`, `-`, `*`, ...
- Allows to define reduction operator for custom DS.

Scheduling & Synchronization

Scheduling

- Static scheduling assigns $N/numCores$ iterations to each core when entering the loop (default)

Listing 6: Scheduling: static (default) vs. dynamic

```
#pragma [...] schedule(dynamic, 5)
for (size_t i = 0; i < N; ++i) {
  // ...
}
```

- Assigns 5 iterations to every idle processor at run-time.

Synchronization

- `#pragma omp critical(NAME)`
- `#pragma omp atomic`
- `#pragma omp barrier`

Example

Listing 7: Synchronization example

```
int value = 0;
bool flag = false;
int N = 100000;
#pragma omp parallel for schedule(dynamic, 1) shared(value)
for (size_t i = 0; i < N; ++i) {
  if (i == 1000)
    N = 10;
  #pragma omp critical
  {
    value += i;
    // do sth. more...
  }
}
printf(      , value);
```

Further reading:

- http://openmp.org/
- https://computing.llnl.gov/tutorials/openMP/
- http://en.wikipedia.org/wiki/OpenMP