AI/ML-Guided Performance Tuning for OpenMP

Vivek Kale
Sandia National Laboratories
SC ‘23 OpenMP Booth Talk
November 10, 2023

“Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.”

SAND2023-84257
Motivating Example Code: MPI-only


```c
#include <mpi.h>
int main(int argc, char* argv[])
{
    MPI_Init(&argc, &argv); // assume a weak scaling where each process gets n+2 elements
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    double* u = malloc(sizeof(double)*n+2);
    double* unew = malloc(sizeof(double)*n+2);
    int tstep = 0;
    while(tstep < 1000){
        if(myrank != 0) {
            MPI_Irecv(leftBoundary, ...);
            MPI_Isend(leftBoundary, ...);
        }
        if (myrank != (numprocs - 1)){
            MPI_Irecv(rightBoundary);
            MPI_Isend(rightBoundary);
        }
        MPI_Waitall(...);
        u[0] = leftBoundary;  u[n+1] = rightBoundary;
        for (i = 0; i < n+2; i++)
            unew[i] = (u[i-1] + u[i] + u[i+1])/3.0;
        MPI_Allreduce(err, MAX);
        tstep++;
    }
    MPI_Finalize();
}
```

Platform: supercomputer, cloud

CC stencil.cpp -lmpich; mpirun -n 4 ./a.out 100 1000;
Slow OpenMP Prohibits MPI Scalability!

Time

Noise delays every timestep on some node\textsuperscript{1,2}

Example: 10 ms timestep, 10 of 1000 timesteps delayed by 5 ms

\begin{itemize}
  \item 1 node
    \begin{itemize}
      \item Noise delay: 50 ms
      \item \textit{Time with noise}: 10.05 seconds $\rightarrow$ 0.5% slower.
    \end{itemize}
  \item 1000 nodes
    \begin{itemize}
      \item Noise delay: $1000 \times (1 - (1.01^{-1000})) \times 5$ ms $= 4449$ ms
      \item \textit{Time with noise}: 14.449 seconds $\rightarrow$ 50% slower!
    \end{itemize}
\end{itemize}

Performance improves if we can perfectly redistribute work within each node\textsuperscript{3}

Noise Amplification problem $\rightarrow$ Open CS problem: Wait-free consensus

\[\text{Wall Clock Time (seconds)}\]

\[\text{Number of MPI Processes}\]

\[\text{Impact of Process-local Perturbance}\]

Perturb Length: 5 ms ; Tp: 10 ms; Timesteps: 1000; Perturb Prob.: 1%

\[\text{Number of MPI Processes}\]

\[\text{Impact of Process-local Perturbance}\]

\[\text{Wall Clock Time (seconds)}\]

\[\text{Number of MPI Processes}\]

\[\text{Impact of Process-local Perturbance}\]

AI/ML to Find Right OpenMP Strategy

• Static scheduling
  + Good locality of data
  - Ignores OS jitter

• Dynamic scheduling
  + Keeps cores busy
  - Poor usage of data locality
  - Can lead to large overhead

• Static + 10% dynamic scheduling

More nodes means higher average delta
→ More nodes means best-performing $f_s$ lower.

```
#pragma omp parallel for schedule(static)
for(int i=0; i<n; i++)
  loop_body(i);
```

```
#pragma omp parallel for schedule(dynamic)
for(int i=0; i<n; i++)
  loop_body(i);
```

```
double fd = predict_dynamic_fraction();
#pragma omp parallel for nowait
for(int i=0; i<ceil((1.0-fd)*n); i++)
  loop_body(i);
#pragma omp parallel for schedule(dynamic)
for(int i=ceil((1.0-fd)*n); i<n; i++)
  loop_body(i);
```

\[
f_s \leq 1 - \frac{\delta_{\text{total}}}{T_p}
\]

AI/ML guided OpenMP Performance Tuning

1. **Generate**: Identify new OpenMP algorithmic strategy’s via defining control points
2. **Tune**: Tune each new strategy’s control points
3. **Search**: find best-performing algorithmic strategy with best-performing parameters

For the above to work, need:

1: Prototyping and experimentation
2: OpenMP features
3: Software tools

---

Token Generation Problems

Bayesian Statistical Models

---

1: Control Points for OpenMP Tuning
CPU-GPU Multi-parameter Tuning

- Different Programming Models
- Tune: {host-device coordination}

Stepwise tuning of asynchronous version of OpenMP offload program

- OpenMP+CUDA
- Tune: {intra-CPU and Intra-GPU} x {host-device coordination} x {host-device load balancing}


Full Node Multi-Objective Multi-Parameter Tuning

- Randomized Mat Mul Summit
- GitHub for task-to-GPU scheduling prototype: github.com/SOLLVE/openmp-rts
- Experiments are done on node of Summit using clang15

Prototype library for the LLVM OpenMP runtime that supports OpenMP task-to-multiGPU scheduling improves performance over OpenMP static by 43.6% and MPI version by 16.8% when affinity scheduling through low-overhead static/dynamic scheduling.

Multi-objective Multi-parameter Tuning: Heat2D

- **Application**: Stencil heat2D with problem size 32768 x 32768, 100 timesteps, Grain size 4
- **Platform**: Spectrum MPI, LLVM OpenMP, use ‘#pragma omp requires unified_shared_memory’, Perlmutter
- **Runtime params**: 1 OpenMP thread per core. For OpenMP multi-GPU versions, used 2 MPI processes per node. cpu to gpu binding.

---

**Static-task**

- CPU [176]
  - CPU 0
  - CPU 1
  - CPU 2
  - CPU 3
  - CPU 4
  - CPU 5
  - CPU 6
  - CPU 7
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CPU [66]
- CPU [1372]
- CPU [176]

**RandomGPU**

- CPU [176]
  - CPU 0
  - CPU 1
  - CPU 2
  - CPU 3
  - CPU 4
  - CPU 5
  - CPU 6
  - CPU 7
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T
- CUDA HW [0004]: 00:00.0 - T

---

More Dynamic-like strategies

2: OpenMP Features for Enabling AI/ML Guided Tuning
OpenMP Set Object: A Way to More Easily Use AI/ML for OpenMP

- Augment OpenMP to be higher-level and consider node-level heterogeneity
- Can consider literature on parallel algorithm design\textsuperscript{10}
- Define sets of entities to describe parallelization of computation in OpenMP:
  1. A set of (OpenMP) work units is distributed across
  2. a set of (OpenMP) devices/threads in way that is
     constrained by a set of dependencies amongst work units and
     aims to satisfy a set of OpenMP associations between work units and/or devices/threads.\textsuperscript{11}

\begin{verbatim}
#pragma omp set(setName) [omp_list_t_kind: {listelemlist_id_1, listelemlist_id_2, listelemlist_id_n}]
\end{verbatim}

\begin{verbatim}
#pragma omp set(myChunkSet) [chunk: {{0, 3}, {4, 7}, {8, 11}, {12, 15}}]
\end{verbatim}

\begin{verbatim}
#pragma omp set(myThreadSet) [thread: {1, 5, 2, 3}]
\end{verbatim}


\textsuperscript{11} https://relate.cs.illinois.edu/course/cs554-f23/slides/slides_02.pdf
Putting it Together: Using Sets in OpenMP for Heterogeneous Programs

```c
#pragma omp parallel num_threads()
#pragma omp single
{
    #pragma omp target spread(levelNum) teams distribute parallel for \
    set(devices:<[device_list]>, <levelNum>) set(chunks: <chunk_list>) spread_schedule(<levelNum>, <strategy>)
    map(close: to: A[omp_spread_start:omp_spread_size] ) \
    map(from: B[omp_spread_start:omp_spread_size])
    depend(in:A[omp_spread_start:omp_spread_size]) nowait
    for (i = 0; i < n; i++)
        doWork(i);
}
```

- `spread_schedule`: sets distribution strategy and chunk size; has all schedules from OpenMP in schedule clause of parallel for and can also take in set of chunk-to-device assignments, which is a user-defined schedule.


3: Automation Tools for AI/ML Guided OpenMP Performance Tuning
From Manual to Automated Performance Improvement

Kokkos User Code

Kokkos::View<double> u(102);
Kokkos::View<double>::HostMirror host_u = create_mirror_view(u);
Kokkos::parallel_for(0, 102, KOKKOS_LAMBDA(int i) {
    unew[i] += (u[i] + u[i-1] + u[i+i])/3.0;
};
swap(u,unew);
Kokkos::deep_copy(host_u, u);

Kokkos Library backends supporting user-defined hints

github.com/vlkale/kokkos

Kokkosp... Callbacks Connector for parameter tuning and measurements

github.com/vlkale/kokkos-tools

Manual, Rose-based clangASTRewriter, ORIO

Binary instrumented Code
- contains Kokkos Tools Callback invocations, which in turn call auto-tuning engine

Performance Optimization Report

Source-to-source code transformation
- based on user-defined hints
- contains any transformation-specific runtime callbacks

./a.out 10000 1000;

Conclusion

- **Challenge**: large search space for OpenMP optimization for MPI+OpenMP
- **Opportunity**: AI/ML can prune large search space significantly
- **Approach**:
  - Training via experimentation of OpenMP strategies, identifying control points
  - Easier machine generation and tuning via OpenMP Set interface.
  - Search for best-performing OpenMP strategies via Kokkos Tools auto-tuning
- **Outcomes**:
  - 8.3x speedup via software tools appropriate control point transformations and tuning on single GPU; 1.7x speedup on multi-GPU experimentation
  - New OpenMP features and software tools for use.
- **Future work**:
  - More investigation of Kokkos Tools auto-tuning with OpenMP+CUDA
  - Experiment with HPX + Kokkos and/or Charm++ + Kokkos
  - Consider full-fledged applications for protein folding and weather simulation
openmp.org

OpenMP API specs, forum, reference guides, and more

link.openmp.org/sc23

OpenMP SC23 booth talk videos and presentations