Task-based scheduling on multi-/many-core architectures

Andra Hugo
andra.hugo@it.uu.se

Post-doc Architecture Research Team
Uppsala University
Parallel Applications Today

• Hungry for performance
  • CPUs, GPUs, XeonPhis, FPGAs…

• Problems
  • Programmability
  • Performance prediction
  • Scalability

• We need software help
Task-based approach

- StarPU, OmpSs, OCR, …
  - Tasks = input + implementation
  - Handles scheduling and data transfers

- Tasks
  - Same input size
  - Sequential or parallel

WHAT GRANULARITY ?

Parallel Applications
- Parallel Compilers
- Parallel Libraries

Task-based Runtime

Drivers (CUDA, OpenCL)

<table>
<thead>
<tr>
<th>CPU</th>
<th>NVIDIA GPU</th>
<th>Intel Xeon Phi</th>
<th>FPGA</th>
</tr>
</thead>
</table>
Task Granularity – CPU vision

• Input data should fit in L1/L2 cache (small & sequential)
  • Good locality
  • Scheduling overhead, and the GPUs?

• Input data should fit in L3 cache (large & sequential)
  • Less scheduling overhead, better for GPUs
  • Less parallelism, L3 shared

• Solution
  • Parallel tasks running alone on a cluster of cores (large & parallel)
CPU cores aggregation

- Software solution to build clusters of cores (StarPU)
  - 2 clusters (10 cores) with L3 30MB each
  - 4 NVIDIA GPUs
- Parallel task (10 threads)
- Sequential task
- Intel MKL implementation
- Task size 30MB

Resource aggregation for task-based Cholesky Factorization on top of heterogeneous machines
HeteroPAR Workshop, joined to EuroPar 2016, Grenoble, France
Task Granularity – GPU vision

• Input data should fit the GDDR5 memory (very large ~5GB & very parallel)
  • Highly parallel tasks with tens of thousands of threads
  • Limits Task Level Parallellism
  • Some tasks still do not scale

Limited inherent parallelism, Too memory bound, Synchronization

• Solution
  • Partition the GPU & co-execute more (smaller & less parallel)
GPU partitioning

- Software solution to partition the GPU (StarPU)
- Co-execute tasks
  - Isolated
  - Fixed allocation of SMs
Scheduler

• Software tools
  • isolate and control
    the parallel tasks

• Minimize total execution time (StarPU)

History based model
Predict execution time of the tasks

- CPU CORE #1
- CPU CORE #2
- CPU CLUSTER #1 (4 cores)
- GPU PARTITION #1 (8 SMs)
- GPU PARTITION #2 (5 SMs)

Scheduled
To be Scheduled
Conclusion

• Task granularity
  = Data input + parallelism

• Software tools
  • Aggregate CPU cores
    • In 1, 2, ..., N clusters
  • Partition GPU devices
    • In 1, 2, ..., (max no SMs) partitions
Difficult problems on multi/many-cores

- Predict the performance of the tasks
  - History based models - large standard deviations
  - One cause: Memory

- Schedule the tasks
  - Applications: too irregular, too large
  - Hardware: too complicated, effort to match applications better
  - Lightweight dynamic online scheduler to choose between all possible configurations for large applications
Task-based scheduling on multi-/many-core architectures

Questions ???

Andra Hugo
andra.hugo@it.uu.se

Please don’t forget them! 😊

Post-doc Architecture Research Team

Uppsala University