A Parallel Programming Course Based on an Execution Time-Energy Consumption Optimization Problem

Javier Cuenca
Department of Technology and Engineering of Computers, University of Murcia, Spain

Domingo Giménez
Department of Computing and Systems, University of Murcia, Spain

EduPar Workshop, IPDPS, Chicago, May 23, 2016
Outline

1. Organization of the course
2. The optimization problem
3. Algorithmic techniques
4. Evaluating Teaching
Course description

Methodology of Parallel Programming

- Fourth year of Computer Science Degree, first semester.
- In the previous semester a course on Parallel Architectures.
- In the second year a course on Principles of Concurrent and Distributed Programming.
- Compulsory for specialization in Software Engineering, optional for specialization in Computer Engineering.
- Medium size class, around 35 students.
- Practical, Problem-based learning approach.
Syllabus

- Review of parallel architectures
- Parallel programming paradigms
- Parallel programming tools: OpenMP and MPI
- Analysis of parallel algorithms
- Methodology of parallel programming
- Parallel algorithmic schemes
Topics of the TCPP Curriculum - Architecture, Cross Cutting

- Review of parallel architectures

<table>
<thead>
<tr>
<th>ARCHITECTURE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture classes</td>
<td>K</td>
</tr>
<tr>
<td>Streams (e.g. GPU)</td>
<td>K</td>
</tr>
<tr>
<td>MIMD</td>
<td>K</td>
</tr>
<tr>
<td>Multithreading</td>
<td>K</td>
</tr>
<tr>
<td>Multicore</td>
<td>K</td>
</tr>
<tr>
<td>Heterogeneous</td>
<td>K</td>
</tr>
<tr>
<td>Shared vs. distributed memory</td>
<td>C</td>
</tr>
<tr>
<td>SMP</td>
<td>K</td>
</tr>
<tr>
<td>NUMA</td>
<td>K</td>
</tr>
<tr>
<td>Message Passing</td>
<td>C</td>
</tr>
<tr>
<td>Cache organization</td>
<td>K</td>
</tr>
<tr>
<td>Impact memory hierarchy on software</td>
<td>C</td>
</tr>
<tr>
<td>Performance Metrics</td>
<td>K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CROSS CUTTING</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Why and what is PDC</td>
<td>K</td>
</tr>
<tr>
<td>Power</td>
<td>K</td>
</tr>
<tr>
<td>Cluster, Cloud, Grid</td>
<td>K</td>
</tr>
</tbody>
</table>
## Topics of the TCPP Curriculum - Programming

<table>
<thead>
<tr>
<th>Programming</th>
<th>Methodology</th>
<th>Tools</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Programming Paradigms</td>
<td>Client server</td>
<td>Language extensions</td>
<td>Performance monitoring</td>
</tr>
<tr>
<td>Shared memory</td>
<td>Producer-consumer</td>
<td>Compiler Directives pragmas</td>
<td>Performance metrics</td>
</tr>
<tr>
<td>Distributed memory</td>
<td>Scheduling and computation</td>
<td>Libraries</td>
<td>Speed-up</td>
</tr>
<tr>
<td>Hybrid</td>
<td>Decomposition strategies</td>
<td>SPMD Notations</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Task/thread spawning</td>
<td>Scheduling and mapping</td>
<td>MPI</td>
<td>Amdahl’s law</td>
</tr>
<tr>
<td>SPMD</td>
<td>Data Distribution</td>
<td>CUDA/OpenCL</td>
<td>Isoefficiency</td>
</tr>
<tr>
<td>Data Parallel</td>
<td>C</td>
<td>C</td>
<td>K</td>
</tr>
<tr>
<td>Parallel loop</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Task and Threads</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Synchronization</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Critical regions</td>
<td>C</td>
<td>C</td>
<td>K</td>
</tr>
<tr>
<td>Deadlocks</td>
<td>C</td>
<td>C</td>
<td>K</td>
</tr>
<tr>
<td>Memory models</td>
<td>K</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
<td><strong>Algorithms</strong></td>
<td><strong>Tools</strong></td>
<td><strong>Algorithms</strong></td>
</tr>
<tr>
<td>Language extensions</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Compiler Directives pragmas</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Libraries</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>SPMD Notations</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>MPI</td>
<td>C</td>
<td>C</td>
<td>K</td>
</tr>
<tr>
<td>CUDA/OpenCL</td>
<td>K</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>
# Topics of the TCPP Curriculum - Algorithms

## Analysis
- Asymptotics cost: C
- Time: C
- Space: C
- Speedup: C
- Cost: C

## Methodology
- Termination detection: C
- Dependencies: C
- Broadcast: C
- Asynchrony: C
- Synchronization: C

## Schemes
- Divide and Conquer: C
- Algorithmic Problems: C
- Matrix computations: K
- Sorting: C
- Graph search: C
- Specialized computations: C
The course is problem-guided.
A bi-objective time execution-energy consumption problem is proposed at the beginning of the course.
Work to be done by the students:

- Presentation on parallelism techniques or tools not in the course syllabus.

- OpenMP and MPI practicals with basic problems. Tools from the Spanish Parallel Programming Contest (luna.inf.um.es). The system evaluates the solutions automatically and in real time.

- Individual tutorials.

- Each student selects a technique to solve the problem. Presentation of the sequential technique selected and parallel algorithmic approaches for the proposed problem.

- Development of parallel versions (shared memory, message-passing and hybrid) and theoretical and experimental analysis.
A master-slave scheme

IN PARALLEL in each process $P_i \ (i = 0, \ldots, p - 1)$ DO

if $i = 0$ then
    for $j = 1$ TO $j = p - 1$ do
        Send task to $P_j$
    end for

    for $j = 1$ TO $j = p - 1$ do
        Receive solution from $P_j$
    end for
else
    Receive task from $P_0$
    Solve task
    Send solution to $P_0$
end if

END PARALLEL
Computational system

- \( p \) processors
- connected through an interconnection network.
- Heterogeneous in:
  - communication
  - computation
  - energy consumption

the costs vary depending on
⇒ the parts of the algorithm and the processor where they are carried out or the source and target processors involved in a communication
⇒ where the master and slave processes are assigned

processes-to-processors mapping problem
Parameters of the system

- **Execution times:**
  - $TimeComunT, \ p \times p$: costs of communications between two processors when sending-receiving a task.
  - $TimeComunS, \ p \times p$: costs of communications between two processors when sending-receiving a solution.
  - $TimeCompuT, \ \text{of size } p$: cost of solving a task by a process in a processor.

- **Energy consumption:**
  - $EnerComunT, \ \text{of size } p$: on a processor working on the communication of a task.
  - $EnerComunS, \ \text{of size } p$: on a processor working on the communication of the solution.
  - $EnerCompuT, \ \text{of size } p$: on a processor when working on the solution of a task.
  - $EnerInac, \ \text{of size } p$: when the processor is idle.
Example

assignation \( \pi = (2, 3, 1, 0) \Rightarrow Time(\pi) = 20, \ Energy(\pi) = 2910 \)
Types of algorithms

- Incomplete exact methods (**Backtracking**, Branch and Bound and Greedy algorithms), with some pruning strategy.
- Distributed metaheuristics (Scatter search, **Genetic algorithms**, Ant colony, Particle swarm swarm optimization, etc.)
- Neighborhood metaheuristics (Hill climbing, **Tabu search**, Guided local search, Variable neighborhood search, Simulated annealing, GRASP, etc.)

Each student works with a method and gives a presentation for the classmates on the application of the method to the optimization problem.
Incomplete Backtracking

- **Sequential:**
  A scheme with a pruning routine. Experiments with different pruning techniques. A maximum number of descendants for each node. A minimum execution time and energy consumption can be associated to a node with a partial assignment. Explore the nodes with the highest estimations. Use of a joint indicator, or a part of the nodes with each objective.

- **Shared-memory:**
  The master generates nodes up to a certain level, starts the slaves, which do backtracking from the nodes dynamically assigned to them.

- **Message-passing:**
  The master sends nodes to the slave processes and these send back the results to the master. Communications if the information about the best solutions is shared. Analysis of the influence of the frequency of communications in the execution time and the Pareto front.

- **Hybrid:**
  Static cyclic distribution among processes and dynamic assignment to the threads in each process.

- Study of speed-up and scalability, and of the influence of the pruning and the sharing of information in the execution time and the Pareto front.
Genetic Algorithm

- **Sequential:**
  Identify population, individual representations and possible forms of the basic routines, and develop a high level genetic scheme.
  Tune the values of the parameters and the routines to the mapping problem.

- **Shared-memory:**
  Independent parallelization of the basic routines.

- **Message-passing:**
  Island scheme. The number of generations to exchange information is one of the parameters to be tuned.

- **Hybrid:**
  An island scheme with threads working on the application of the basic functions at each island.

- **Study of speed-up and scalability, and of the influence of some parameters (number of islands, size of generations and migrations, etc)** on the execution time and the Pareto front.
Tabu Search

- **Sequential:**
  Identify set and element representations and possible forms of the basic routines, and develop a high level Tabu search.
  Tune the values of the parameters and the routines to the mapping problem.

- **Shared-memory:**
  Select a number of nodes to explore at each step depending on the number of threads, or conduct several independent searches.

- **Message-passing:**
  Independent searches at each process, with different initial solutions and search strategies.

- **Hybrid:**
  Independent search in each thread.

- **Study of speed-up and scalability, and of the influence of knowledge sharing on the execution time and the Pareto front.**
<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session on techniques and tools interesting and should be maintained</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The use of the PPC helps to learn basic concepts of parallel programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session on presentation of the methods useful to compare approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem helps to understand energy aware algorithms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem helps to understand heterogeneous systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem helps to understand bi-objective optimization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem helps to understand heuristics and metaheuristics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem was useful for guiding the course</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The mapping problem motivated study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working with the problem made the course more difficult</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The students found the organization of the course appropriate.
The problem selected to guide the course was useful for a deeper learning of concepts previously studied.
Working with a challenging problem is perceived as interesting.
The students consider the organization of the course did not suppose an important increase in their work load.
A Parallel Programming Course Based on an Execution Time-Energy Consumption Optimization Problem

Javier Cuenca
Department of Technology and Engineering of Computers, University of Murcia, Spain

Domingo Giménez
Department of Computing and Systems, University of Murcia, Spain

EduPar Workshop, IPDPS, Chicago, May 23, 2016

Questions? Comments? Recommendations?