Early Adopter - UNRC - Río Cuarto, Argentina

PDC topics in new curriculum proposal for Lic. in Computer Science

Marcelo Arroyo, Nazareno Aguirre, Marcela Daniele

Dpto. de Computación - FCEFQyN - Universidad Nacional de Río Cuarto - Argentina

EduPar 2011
Our current curriculum and PDC topics

Actual curriculum courses and PDC topics coverage

1. **Architecture topics**: Taxonomy, buses, ...
   - Computer architecture (2\textsuperscript{nd} year)
   - Operating systems (5\textsuperscript{th} year)

2. **Programming topics**: Concurrency. Data races. Synchronization. Shared memory and message passing models, ...
   - Programming Languages and Paradigms (3\textsuperscript{rd} year):
   - Operating systems (5\textsuperscript{th} year)
   - Telecommunications and Distributed Systems (5\textsuperscript{th} year)
   - Parallel programming ((elective) 4\textsuperscript{th}/5\textsuperscript{th} year)

3. **Algorithmic topics**:  
   - Programming Languages and Paradigms (3\textsuperscript{rd} year)  
   - Parallel programming ((elective) 4\textsuperscript{th}/5\textsuperscript{th} year)
### Problems with our current curriculum

#### Architecture topics
- Basic coverage *Computer Architectures* course. Not coverage of SIMD instructions, cache, ...  
- *Operating Systems* course focuses on uniprocessor systems

#### Programming
- Basic multithreading practice (using Java, Oz, ...)
- Basic coverage and practice about performance

#### Algorithmic
- Very basic coverage of parallel algorithmic

#### Other PDC topics
Covered only in an *elective* course (in 4\text{th}/5\text{th} year)

---

Marcelo Arroyo, Nazareno Aguirre, Marcela Daniele

Early Adopter - UNRC - Río Cuarto, Argentina
New curriculum proposed

Moving and improving existing courses
1. Down the *Operating Systems* course to 2\(^{nd}\) year
2. Deeper topics on *Computer Architecture* course (e.g. SIMD, MIMD architectures, \ldots)

Introducing new courses
1. *Data Structures and Algorithms III* (*3\(^{rd}\) year*) which includes advanced data structures and design and implementation of concurrent/parallel algorithms
2. *Theory of Computation* (*4\(^{th}\) year*). It will cover foundations of concurrency and modeling concurrent and distributed systems

Teaching concurrency, parallel and distributed topics
**Challenge**: Provide a natural and coherent view of concurrency/parallelism in early learning stage (*2\(^{nd}\) and 3\(^{rd}\) year*)

Marcelo Arroyo, Nazareno Aguirre, Marcela Daniele  
Early Adopter - UNRC - Río Cuarto, Argentina
Including PDC practice in other existing courses

### Candidate courses

- **Math**
  - Linear algebra
  - Numerical methods
  - Geometry
- **Computer Graphics and Animation**
- **Simulation**: Cellular Automata, Parallel DEVS, . . .
- **Compilers**: CPU+GPU code generation
- **Validation and Verification of Software**: methods and tools for verification (safety, liveness, fairness, race conditions, deadlocks, . . .)
- **Computability and complexity**
New curriculum proposed (cont.)

Improving the *Parallel Programming and Distributed Systems* course

Early introduction of topics on concurrent-parallel and distributed systems allow us to improve the course with deeper coverage of PDC:

Contents:

- Parallel computing architectures: Flynn taxonomy, memory organization, ...
- Programming models: Shared memory model, message passing model. Synchronization, ...
- Performance analysis: parallel computing models, algorithm complexity, ...
- Design of parallel algorithms: Decomposition techniques, task graphs, communication patterns, mapping, ...
- Practice: pthreads, OpenMP, MPI, Erlang, ...
New curriculum proposed (cont.)

Improving the *Parallel Programming and Distributed Systems* course: New topics

- Inclusion of SIMD instruction sets: SSE-Altivec
- Inclusion of modern languages, frameworks and libraries (e.g. Intel TBB, Pfunc, ...)
- Models and tools for profiling to improving performance
- Models and tools for race conditions and deadlocks detection
- CPU+GPU architectures and programming (CUDA, OpenCL)

New teaching strategies

- Adopting some ideas from Intel’s proposal for teaching parallelism using games
  - Multicore game engines introduction
  - Practice: development of parallel version of the *Destroy the Castle* game
We have a *curriculum committee* to follow the curriculum development.

We’ll instruct the committee to include evaluation actions to follow the evolution of parallel learning issues.

In particular:
- Conduct tests and surveys students at end of each course
- Surveying teachers who require some PDC issues
- Periodically comparing the coverage of our curriculum topics with the TCPP curriculum

**Students evaluation on PDC topics**
- Regular tests covering theory and practice
- Homeworks and laboratory projects
We think . . .

- The TCPP proposal is very useful to help us to build a modern curriculum.
- Our actual *Parallel Programming* course is close to proposed TCPP course.
- We have the opportunity to change our curriculum, so we want to introduce PDC topics across it.
- The new curriculum will be close to the TCPP proposal in terms of coverage of PDC topics.
- We will need to make adjustments periodically.
- We will need to work collaboratively to improve the proposal.
- It will be very useful to share materials and tools.
Questions?