How to avoid making the same Mistakes all over again: 
What the parallel-processing Community has (failed) to offer the 
multi/many-core Generation

Jesper Larsson Träff 
NEC Laboratories Europe, NEC Europe Ltd., St. Augustin, Germany 
traff@it.neclab.eu

1 Context and Observations

We observe that in the past decade parallel processing and parallel algorithmics have disappeared from mainstream computer-science curricula and moved either into advanced graduate courses or into the application domains. This is well illustrated by current textbook availability. The influential book by Cormen, Leiserson and Rivest [1] in its first edition had a substantial chapter on PRAM algorithmics that was dropped from the second edition [2], the parallel algorithms book by JáJá [4] is no longer in print, and recent algorithms texts by Kleinberg and Tardos [5], or Dasgupta et al. [3] do not touch on parallelism at all. In the past decade it was well possible to complete an advanced computer science degree without exposure to parallel processing, and in particular parallel algorithms.

This may be appraised by some as a good sign: Parallel processing is basically a set of specialized problem-solving techniques, and is rightly taught as a side-issue to “scientific computing” with its concern for problems with very large data sets. It is not a fundamental discipline, and therefore does not belong in a core computer-science curriculum.

On the other hand: With the advent of multi-core processors, and the industry-wide expectation that the number of cores on a chip will increase exponentially over the next years, with currently little or no increase in sequential performance in sight, parallelism, be it implicit or explicit, will become fundamental in all application areas where performance matters. The demand for skilled experts with solid foundations can be expected to increase accordingly.

Thus, a major discrepancy between what computer-science students have been and are being taught and what will be required in the next years may be looming. In the short run the demand will be met with candidates with little exposure to parallel-processing techniques, or with an inadequate, limited perspective. This could of course lead to refreshingly new ways of looking at the emerging problems, but equally well to a waste of time in repeating old mistakes, and reinventing techniques and paradigms that have already been proved in the past. The current debate about transactional memory, threads [6], and other examples of either evasion or recycling of the issues, may be indications of the latter happening.

It is timely for the parallel-processing community to take stock: What does the community have to offer the upcoming generation that will have to deal with parallelism for a much broader range of applications? What are the fundamental paradigms and techniques of the past? How can these be most effectively conveyed, and to whom? Which were the mistakes and wrong turns of the past? How can repetition be avoided? Which problems remain unsolved, and what are the major, new challenges?

2 The Questions to the Panelists

Q1: Are the observations correct? Is there a gap in computer-science education?

Q2: If yes, what would be the worst consequences?

Q3: If any, what are the most painful mistakes already being made? Could these have been avoided? Did a better solution already exist?

Q4: If “no” to Q1, why not? Are things fine?

Q5: What does the parallel-processing community have to offer the multi-core generation? What are the key techniques, insights and paradigms that the next generation of parallel-software developers should be taught?

Q6: How should this be done? What should computer-science education in parallel processing look like? Should it be taught at all?

Q7: What will the multi-core generation need that the parallel-processing community has not yet addressed? Can help be expected from other sides?

Q8: What are the skills that industry is currently missing the most?
References


