Prof. Dr. C. Stamm christoph.stamm@fhnw.ch Tel.: 056 202 78 32

Exercise 3

1 Cost-Optimality and Minimum Execution Times

Consider a parallel system with *p* processing elements solving a problem consisting of *W* units of work in time $T_p = \frac{\sqrt{W}}{2} + \frac{W}{p}$.

a) Compute the cost, overhead, speedup, efficiency, and isoefficiency for the parallel system.
b) Is the parallel system scalable?
c) Can the problem be solved cost-optimally for $p = \Theta(W)$?
d) Assume $p < \Theta(W)$, can now the problem be solved cost-optimally? If yes, what is a valid upper bound for *p*?
e) Let $C(W) = ½W$ be the degree of concurrency. Compute the minimum execution time.
f) Compute the lower bound on the parallel runtime for solving the problem cost-optimally.

2 Cost-Optimality and Isoefficiency

Consider a parallel system with *p* processing elements solving a problem consisting of *W* units of work. Prove that if the isoefficiency function of the system is worse (greater) than $\Theta(p)$, then the problem cannot be solved cost-optimally with $p = \Theta(W)$. Also prove the converse that if the problem can be solved cost-optimally only for $p < \Theta(W)$, then the isoefficiency function of the parallel system is worse than linear.

3 Vector Operations

In Exercise 2 you programmed an OpenCL kernel `edges(…)`. Because the edge detection is concurrently computed on three image channels, this kernel offers several possibilities to use built-in math and vector operations. Rewrite the kernel code by using built-in math and vector operations.

4 GPU and CPU

In Exercise 2 the source file “ocl.cpp” contains the procedure `initOCL(…)`, which builds the kernel for the first found GPU device. Depending on your hardware the procedure `initOCL(…)` enumerates several different platforms and/or GPU devices. Parameterizing the device type (`CL_DEVICE_TYPE_GPU` or `CL_DEVICE_TYPE_CPU`) in the same procedure would allow building your kernel either for a GPU or a CPU.

Extend the test program so that the OpenCL edge detection kernel is started twice, once on GPU and once on CPU. What is the difference in performance compared to the OpenMP implementation? How do the resulting images differ?

Hints: In case OpenCL only lists one platform with one GPU device, then you have to install an additional OpenCL driver for your CPU. Intel provides both an OpenCL driver and an additional OpenCL SDK for Visual Studio: http://software.intel.com/en-us/vcsource/tools/opencl-sdk. The OpenCL SDK contains an OpenCL syntax checker, pre-compiler, and debugger. To invoke the debugger you have to build the kernel with additional parameters and then to start the debugger on the OpenCL kernel on your CPU. The additional parameters are:

```
program.bild(devices, “-g –s \”full-path\kernel-file-name\””)
```