

Advanced Computer Architectures

ACA319

Lab Assignment 1

week 3

MESSAGE PING-PONG

EXPERIMENT 1

Write a C program using the **MPI** library with two processes, A and B.

In a continuous loop, A has to send a message to B, then after receiving the message, B has to send a message to A, and so on.

Whenever B receives a message, print "**ping**" to the screen, whenever "A" receives a message, print "**pong**" to the screen.

EXPERIMENT 2

Delete the screen printouts from Experiment 1 and measure the timing for 1,000 message pairs (sending from A to B and back to A) for:

- (a) message size = 1 Byte
- (b) message size = 1024 Bytes
- (c) message size = 128 KB

Use message sizes from (b) and (c) to calculate the overall transfer rate in MB/s, use message size from (a) to estimate communication latency.

Advanced Computer Architectures

ACA319

Lab Assignment 2

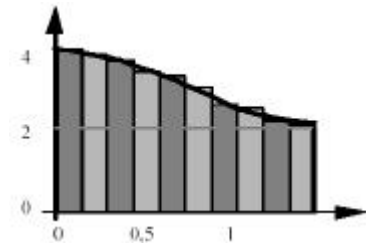
week 4

PI APPROXIMATION BY INTEGRATION

EXPERIMENT 1

Write a C program using the **MPI** library for approximating Pi by using the following formula:

$$\pi = \int_0^1 \frac{4}{1+x^2} dx = \sum_{i=1}^{\text{intervals}} \frac{4}{1 + ((i - 0.5) * \text{width})^2} * \text{width}$$



The parallel program structure is to be constructed in the following way:

- Variable number of worker processes
- Each worker process loads a block of 1,000 segments to be calculated at a time from a common master
- Each worker reports back its final result (sum of all its segments) to the master
- The master prints the total result (compare with built-in value of π)
- Use MPI event logging to **monitor the performance** of your system.

Advanced Computer Architectures

ACA319

Lab Assignment 3

week 5

MANDELBROT SET

EXPERIMENT 1

Write a C program using the **MPI** library calculating the Mandelbrot set.
For each point c in the complex domain calculate the following iteration:

$$z_0 := 0$$
$$z_{i+1} := z_i^2 + c$$

Remember:

- Complex addition: $(a,b) + (c,d) = (a+c, b+d)$
- Complex multiplication: $(a,b) * (c,d) = (a*c - b*d, a*d + b*c)$
- Complex magnitude: $\text{mag}(a,b) = \sqrt{a^2 + b^2}$

Compute an array of 50x50 pixels with the image area:

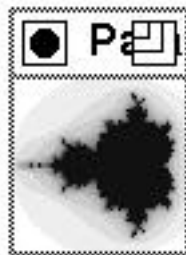
$$\begin{array}{l} \text{real part} \quad [-2.0, +1.0] \\ \text{imaginary part} \quad [-1.4, +1.4] \end{array}$$

The iteration for a point should stop when either $\text{mag}(z) > 2.0$ or 100 iteration steps have been executed (whichever comes first).

The parallel program structure is to be constructed in the following way:

- Variable number of worker processes
- Each worker process loads the next area to be calculated from a common master
- Each worker reports back its final results (point values) to the master
- The master prints out the result array.

This can be done as character graphics (print "X" for value 100, print "_" else)
or as pixel graphics (**bonus point**)



Advanced Computer Architectures

ACA319

Lab Assignment 4

week 6

TEMPERATURE GRADIENT

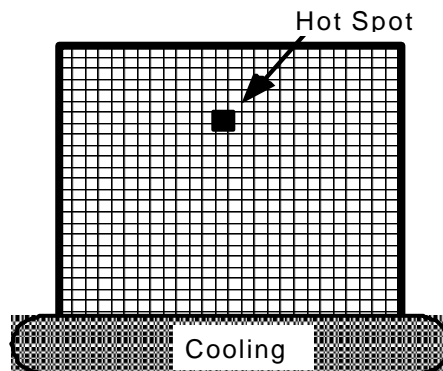
EXPERIMENT 1

Write a **sequential** C program for simulating a temperature gradient distribution of a 50x50 area. Temperatures are represented by integer values between 0 and 255.

Assume a hot spot at position [15, 25], which always has a temperature of 255 (heater element). All other cells have room temperature (23), all elements in the bottom line [49, *] are being cooled with ice (constant temperature 0).

For every time step, each cell calculates the average value from its four neighbors:

$$t_{i,j}^{n+1} := (t_{i-1,j}^n + t_{i,j-1}^n + t_{i+1,j}^n + t_{i,j+1}^n) / 4$$



Save the calculated temperature gradient for every (or every 10th) time step as a PPM graphics file for 1,000 iterations.

EXPERIMENT 2

Write a C program using **threads** to solve the temperature gradient distribution.

Bonus point: Combine the individual images generated into an MPG animation.

Advanced Computer Architectures

ACA319

Lab Assignment 5

week 7

TEMPERATURE GRADIENT

EXPERIMENT 1

Write a parallel C program using the **MPI** library for implementing the temperature gradient problem from the previous lab assignment.

The parallel program structure is to be constructed in the following way:

- Variable number of worker processes
- Each worker processes a fixed area of the temperature field (no communication required for this part of the problem)
- Each worker reports back its final results (point values).

Only the result array is being printed (after 1,000 iterations - no intermediate results).

Bonus point: Use asynchronous communication overlapping processing.

EXPERIMENT 2

Compare both the run-time (speedup) and the result (correctness) with the sequential version and the pthread version from the previous lab assignment.

Advanced Computer Architectures

ACA319

Lab Assignment 6

week 8

PARALLEL SORTING

EXPERIMENT 1

Write a C program using the **MPI** library implementing a parallel version of bucket sort.

- Each process receives an equal number of elements and groups them into buckets.
- Initially each process sets up as many buckets as there are processes in total
- After an initial bucket exchange, each process only maintains a single bucket.
- Each process sorts its local bucket.
- Combine buckets from all processes for final solution (write to text file).

EXPERIMENT 2

Re-run experiment 1 with only a single task and record the average time for the 5 unsorted sample files. Calculate the speed-up.

EXPERIMENT 2

Run the Unix **sort** program to sort the 5 unsorted sample files and record the average time. Calculate the speed-up compared to this algorithm.

Advanced Computer Architectures

ACA319

Lab Assignment 7

week 9

GENETIC ALGORITHMS

EXPERIMENT 1

Write a C program using **pthread**s library that optimizes a problem solution using genetic algorithms.

Genetic algorithms (GA) are one technique for finding approximations for compute-intensive problems. The idea of GAs follows the evolutionary principles of life.

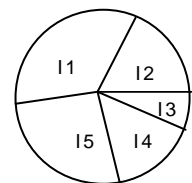
A GA simulation runs over a number of **generations**. In each generation, we keep a set of **genes** (genotypes or set of codes), from which we construct **individuals** (phenotypes), which can be evaluated by means of a fitness **function**.

In our simple case, we want to run a GA with 20 genes over 500 iterations. Each gene is a bitstring (8 bits) and we want to maximize the function x^2 . E.g.:

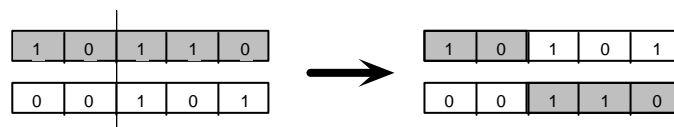
$$\text{fitness}(0001\ 0110) = (0+0+0+16+0+4+2+0)^2 = 22^2 = \mathbf{484}$$

In every time step, the best individuals are selected for reproduction:

(a) Use a random selection, biased by the fitness of each individual.
I.e. individuals with higher fitness are more likely to be selected (maybe even more than once), as indicated graphically by the "wheel of fitness"



(b) Use cross-over at a random position (1..7) for pairing up selected individuals



(c) Use mutation with a very low probability only.

Mutation flips a single bit at a random position in one of the newly generated individuals.

The algorithm stops when either a individual with sufficiently high performance has been produced (e.g. 90% of max. possible), or a max. number of iterations has been reached.

Advanced Computer Architectures

ACA319

Lab Assignment 8

week 10

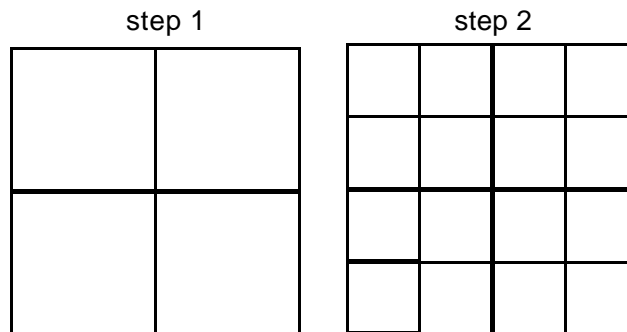
FRACTAL MOUNTAINS

EXPERIMENT 1

Write a SIMD/SPMD program for creating a 2D fractal mountain surface using midpoint-replacement in 2D (see 1D version in figure below).



At each iteration step, adjust the mid-points of all 2D patches by a weighted random value up/down. For every step, each patch will be recursively subdivided into 4 smaller patches, until the patch size reaches a single pixel (image resolution).



Solve this problem with $4^5 = 1024$ parallel processes, emulating a quadtree structure. Print the generated elevation data either as a text file with height values or generate a rendered image file (bonus point).

Advanced Computer Architectures

ACA319

Lab Assignment 9

week 11

IMAGE PROCESSING

We are given an original image and a image with noise overlay
For the following experiments, calculate the results of the image functions and compare to the original by using the average **sum of squared difference** over all pixels.

$$\text{Error} = 1/(n*m) * \sum_{i,j} (\text{original}_{i,j} - \text{noiseimage}_{i,j})^2$$



EXPERIMENT 1

In SIMD parallel, calculate the minimum function in a 3x3 neighborhood

EXPERIMENT 2

In SIMD parallel, calculate the maximum function in a 3x3 neighborhood

EXPERIMENT 3

In SIMD parallel, calculate the average function in a 3x3 neighborhood

EXPERIMENT 4

In SIMD parallel, calculate the median function in a 3x3 neighborhood (middle element in sorted sequence))

1	10	5
2	4	15
6	5	3

- a) Min.: 1
- b) Max.: 15
- c) Avg.: 6 (rounded)
- d) Median: 5

Advanced Computer Architectures

ACA319

Lab Assignment 10

week 12

TRAFFIC SIMULATION

EXPERIMENT 1

Implement a traffic simulation for a single, looped lane as a SIMD/SPMD parallel program. Assign one PE to each car and simulate the traffic situation over a number of iterations. Vary the number of cars until spontaneous traffic jams occur.

At the end of each iteration, plot any standing car (traffic jam!) along a line representing the length of the road.

Each car has to follow the same rules:

- Cars start at rest in equal distance, accelerate from 0
- Cars brake if distance in front too short (add some random factor)
- Cars accelerate if distance in front is sufficient (add some random factor)
- Assume a fixed max. acceleration
- Assume a fixed max. speed (speed limit)

A bonus point is awarded for advanced graphics output of the solution.



Advanced Computer Architectures

ACA319

Lab Assignment (Supplementary)

week X

STEREO VISION

EXPERIMENT 1

Implement a SIMD/SPMD parallel program in C/MPI for **generating** a random dot stereogram. A subroutine shall allow the lifting of rectangular areas above the drawing level. The output shall be two binary images (shown as red/green stereogram below).

EXPERIMENT 2

Implement a SIMD/SPMD parallel program in C/MPI for **analyzing** a random dot stereogram. The subroutine shall take left and right stereo image and return an elevation map as a grayscale image (see below).

A bonus point is awarded for advanced graphics output of the solution.

