EXAM

DESIGN AND ANALYSIS OF ALGORITHMS
FOR PARALLEL COMPUTER SYSTEMS
9 – 15 AT ÖP7

Maximum number of points is 60. For the grade 3 you need 30 points, for the
grade 4 you need 40 points and for the grade 5 you need 50 points. The problems
are in no particular order.
Allowed aids:

• Pocket calculator.

NB!
START EACH PROBLEM ON A NEW PAGE AND WRITE YOUR
CODE NUMBER ON EACH PAGE. CAREFULLY EXPLAIN YOUR
REASONING. ANSWER YOUR QUESTIONS IN ENGLISH AND/OR
SWEDISH.

Good Luck!
Problem 1: \((10=4+4+2)\) Performance and Scalability

![Profile of a parallel application executing on eight processors.](image)

Figure 1: Profile of a parallel application executing on eight processors.

a. (4) Consider the profile in Figure 1. Answer the following questions, assuming that the serial execution time is \(T_s = 120\).

(i) What is the parallel execution time, \(T_p\), and parallel efficiency, \(E_p\), of the execution shown in the figure?

(ii) What is the total parallel overhead?

(iii) How much total excess computation does the parallel application perform?

(iv) Assuming you have answered the questions above correctly, you may have noticed that there is something exceptional about one of your answers. Give an example of a practical cause of such results.

b. (4) Explain what the isoefficiency function says about the scalability of a parallel system. How can you use the isoefficiency function to choose a suitable problem size when scaling up to a large number of processors?

c. (2) The results of experiments on several different problem sizes and numbers of processors are summarized in the following table.
<table>
<thead>
<tr>
<th>$W$</th>
<th>$p = 16$</th>
<th>$p = 32$</th>
<th>$p = 64$</th>
<th>$p = 128$</th>
</tr>
</thead>
<tbody>
<tr>
<td>625</td>
<td>0.60976</td>
<td>0.43860</td>
<td>0.28090</td>
<td>0.16340</td>
</tr>
<tr>
<td>2500</td>
<td>0.75758</td>
<td>0.60976</td>
<td>0.43860</td>
<td>0.28090</td>
</tr>
<tr>
<td>5625</td>
<td>0.82418</td>
<td>0.70093</td>
<td>0.53957</td>
<td>0.36946</td>
</tr>
<tr>
<td>10000</td>
<td>0.86207</td>
<td>0.75758</td>
<td>0.60976</td>
<td>0.43860</td>
</tr>
<tr>
<td>15625</td>
<td>0.88652</td>
<td>0.79618</td>
<td>0.66138</td>
<td>0.49407</td>
</tr>
<tr>
<td>22500</td>
<td>0.90361</td>
<td>0.82418</td>
<td>0.70093</td>
<td>0.53957</td>
</tr>
<tr>
<td>30625</td>
<td>0.91623</td>
<td>0.84541</td>
<td>0.73222</td>
<td>0.57756</td>
</tr>
<tr>
<td>40000</td>
<td>0.92593</td>
<td>0.86207</td>
<td>0.75758</td>
<td>0.60976</td>
</tr>
</tbody>
</table>

Each cell contains the parallel efficiency observed on the corresponding problem size and number of processors. Make an intelligent guess of the system’s asymptotic isoefficiency function and motivate your answer. You may assume that it is polynomial.

**Problem 2: (7=4+3) Communication Operations**

a. (4) Consider a parallel computer with a $\sqrt{p} \times \sqrt{p}$ two-dimensional torus (wrap-around mesh) network with cut-through routing, i.e., no per-hop delays. Construct an algorithm for the all-to-k reduction operation, where the end result should be present on each of $1 \leq k \leq \sqrt{p}$ processors. Assume that the $k$ processors are on different rows in the torus/mesh. Present your algorithm in plain English or Swedish but be accurate and do not omit crucial details.

b. (3) Find the parallel execution time, $T_p$, of your algorithm. How does the execution time, in general, relate to the standard all-to-one reduction operation (on a two-dimensional torus)?

**Problem 3: (10=4+6) Parallel Algorithm Design Principles**

a. (4) A serial algorithm is often described as a sequence of steps to solve a given problem. State four more issues that must be resolved during non-trivial design of parallel algorithms. Give examples to illustrate.

b. (6) Below is a list of three fundamental parallel algorithm models. Explain each of the models and describe how locality is preserved in the models.

(i) The data-parallel model.
(ii) The task graph model.
(iii) The work pool model.

**Problem 4: (10=6+2+2) Dense Matrix Computations**

a. (6) Describe how one typically tries to arrange the computations (related to dense matrices) to utilize the memory hierarchy in the best possible way. Define the concepts temporal locality and spatial locality. Explain how recursive algorithms/matrix storage formats can impact positively on temporal/spatial locality.
b. (2) The parallel numerical library ScaLAPACK relies on the two-dimensional block cyclic layout to store dense matrices. Explain the tradeoffs that motivate this particular choice of data layout.

c. (2) Visualize a $5 \times 7$ matrix, partitioned into blocks of size $2 \times 2$, and distributed onto a $2 \times 2$ processor mesh by labeling each matrix element with the processor that the element is mapped to by the two-dimensional block cyclic layout.

Problem 5: (8=4+4) SIMD and Multicore

Physics simulations lead to large, sparse linear systems of equations that are typically solved with iterative methods. A fundamental and computationally intensive operation in iterative methods is the sparse matrix-vector product $y \leftarrow Ax$, where $A$ is an $m \times n$ sparse matrix, i.e., $A$ has far fewer than $O(mn)$ nonzero elements. To obtain high performance, implementations tend to exploit SIMD units and multithreading. As you know, parallelism added as an afterthought to an existing sequential program often results in suboptimal performance. In this case, the storage format of the sparse matrix $A$ plays a very important role.

In this problem, you will analyze the pros and cons of two candidate sparse storage formats when it comes to implementing a sparse matrix-vector product on a multicore architecture with one SIMD unit per core. Note that parallelism should be exploited on two levels: thread level and SIMD level. You need to address the issues of decomposition, thread synchronization, data reordering, vector reduction, and memory access patterns such as contiguous access (stride 1) and gather/scatter access. A “good algorithm” on the thread level requires little or no thread synchronization and has a decomposition that balances the load evenly. On the SIMD level, a “good algorithm” requires little or no data reordering and vector reductions and has a contiguous access pattern.

a. (4) The Compressed Sparse Row (CSR) format stores the nonzeros of each row contiguously, one row after the other. For example, the elements in the matrix $A$ below are labeled according to the order in which they are stored in memory. Dashed ($-$) elements correspond to zero elements and are not stored.

\[
A = \begin{bmatrix}
1 & - & 2 & - & - & - & 3 & - \\
- & 4 & - & - & 5 & - & - \\
- & - & 6 & 7 & 8 & - & - \\
- & 9 & 10 & - & - & 11 & - \\
- & - & 12 & - & 13 & - & - \\
14 & - & 15 & - & - & 16 & - \\
- & 17 & - & 18 & - & - & - \\
- & - & 19 & - & - & 20 & - 
\end{bmatrix}
\]

Design and present a good algorithm for the sparse matrix-vector product operation. Use plain English or Swedish and discuss the pros and cons of your algorithm. You may assume that once you have read an element of $A$ you also know where in $A$ it belongs, i.e., its row $i$ and column $j$.

b. (4) The jagged diagonal storage format stores the first nonzero element of each row contiguously. Then the second nonzero element of each row are stored contiguously, and so on. For example, the elements in the matrix
below are labeled according to the order in which they are stored in memory. Dashed (−) elements correspond to zero elements and are not stored.

\[
A = \begin{bmatrix}
1 & - & 9 & - & - & - & 17 & - \\
- & 2 & - & - & 10 & - & - & - \\
- & - & - & 3 & - & 11 & 18 & - \\
- & 4 & 12 & - & - & - & 19 & - \\
- & - & 5 & - & - & 13 & - & - \\
6 & - & - & 14 & - & - & - & 20 \\
- & 7 & - & - & 15 & - & - & - \\
- & - & 8 & - & - & - & 16 & - \\
\end{bmatrix}
\]

Design and present a good algorithm for the sparse matrix-vector product operation. Use plain English or Swedish and discuss the pros and cons of your algorithm. You may assume that once you have read an element of \( A \) you also know where in \( A \) it belongs, i.e., its row \( i \) and column \( j \).

Problem 6: (7=4+3) Graph Algorithms

a. (4) The connected components of an undirected graph \( G = (V, E) \), are the maximal disjoint sets \( C_1, C_2, \ldots, C_k \), such that \( V = C_1 \cup C_2 \cup \ldots \cup C_k \), and \( u, v \in C_i \) if and only if \( u \) is reachable from \( v \) and \( v \) is reachable from \( u \). Formulate a high-level parallel algorithm for finding the connected components of an undirected graph using a 2-D block mapping.

b. (3) Derive asymptotic expressions for the communication time as well as the computation time. If the isoefficiency function exists, then derive it, otherwise explain why it does not exist.

Problem 7: (8=2+3+3) Search Algorithms for DOPs

a. (2) Explain why you sometimes observe superlinear speedup in search algorithms for discrete optimization problems. For which type of search algorithm would you be least surprised to see superlinear speedups and why? The two types to consider are depth-first search and best-first search.

b. (3) Describe the issues surrounding parallelization of the iterative-deepening \( A^* \) algorithm (IDA* algorithm). Recall that IDA* is a path length limited algorithm of the depth-first search type where the path length limit is increased iteratively.

c. (3) Describe Dijkstra’s token termination detection algorithm. Consider the total parallel overhead due to the termination algorithm in the case where all processors finish at the same time. Answer with an asymptotic bound.