Uppsala University

Department of Computer Systems (DoCS)

Repeat Examination

Jan 2000

IT3 (Datorsystem II : (Del 2))

Distribuerade och Realtid System

INSTRUCTIONS TO CANDIDATES

- This is a FIVE (5) hour examination
- Answer all questions in English or Swedish
- Dictionaries are Permitted
- The exam has 10 questions on 5 pages
- Each section 50 points, 30 pts to pass in each section.

Section A : Parallel and Distributed Systems

Question 1

List the major differences between shared and distributed data models of programming. Give an example of a distributed data program and a shared data program.

[8]

Question 2

Explain the features of one programming language that you have studied that implements a facility to provide asymetric, context addressed, asynchronous communication.

[4]

Question 3

Distinguish between data partitioning and load balancing. What are they, and why is each of these things important in achieving high performance programs?

[6]

Give an example application where even division of the data among the PE's does not give good load balancing.

[4]

Outline an approach to load balancing which solves the load balancing problem while retaining the necessary properties of data partitioning.

[4]

Question 4

Give an overview of the design and limitations of the Manchester dataflow computer architecture.

[6]

Question 5

What does the acronym MIMD stand for in the Flynn taxonomy of parallel systems? Show how this class of computers can be specialised into additional sub-categories, and explain the basis for the categories you have presented.

[4]

Question 6

Assume that you can create 1..N parallel copies of a C function using the command $ProcName(1..N, < params >)$; Using this C like parallel programming environment write a parallel function to search through all the values in an array A to find the value which occurs most frequently.
Your algorithm should be able to calculate how to divide up the data evenly among the k copies of the function for any value of k in the range [1..N/2] You can assume that both N and k are powers of 2, that is $N = 2^a$ and $k = 2^b$.
You also need to think about how the copies of the function can know when to stop executing. Explain your termination condition.
Each procedure copy can assume the following interface.

void FindFreq(int Id, int A[ ], int N, int k);

Where, Id is the identity of the copy of the procedure, A is the Array of values to be searched, N is the number of entries in the array, k is and the number of processes.

[14]

Section B : Realtime Systems

Question 7

What system characteristics do we need to know to devise flexible realtime schedules? List the things we need, and explain in a single paragraph why they are needed.

[6]

Question 8

Describe the data structures and activities associated with implementing a tick driven scheduler. Using these structures and activities reason about the delays experienced by a task due to the activity of the scheduler and extend the following equation to include the costs associated with the scheduler execution costs.
You can assume the following equation as a starting point.

$$R_i = J_i + \omega_i$$

$$\omega_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{\omega_i + J_j}{T_j} \right\rceil C_j$$

[10]

Question 9

Define each of the following.

- Deadline monotonic
- Utilization bound
- Schedulability

[6]

Question 10

In a system where blocking can occur, schedules using standard priority assignment and worst case response time calculations can incorrectly predict that a system is schedulable.

(a). Specify deadlines, periods, computation times and resource sharing activity for a system of three tasks. Use a task vs time diagram to show how blocking on the shared resource combined with standard scheduling causes a priority inversion and thus the high priority task to misses its deadline.

Compute worst case response times for your system of tasks using the standard analysis for deadline monotonic priority assignment without blocking delays. Show that the system of tasks you have proposed is predicted to be schedulable by this analysis.

[10]

(b). Describe in detail the approach for dealing with priority inheritance that resolves the problem of priority inversion. Using the same tasks set you defined in (a). draw a picture that shows how applying the priority handling approach you have described resolves the priority inversion problem and allows all tasks to meet their deadlines in the worst case situation.

[10]

(c). Finally, use the following equation to compute the worst case response times for the task set you have proposed using the priority handling approach you described in (c). Compute the blocking delay for each task, and then the worst case for each task. Thus, show that your system of tasks is provably schedulable in all cases.

You will need the following formula,

$$R_i = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where the blocking factor is defined by

$$B_i = \max_{\{s,t \mid t \in lp(i) \wedge s \in used\_by(t) \wedge ceil(s) \geq pri(i)\}} (cs_{t,s})$$

[8]