

THE AUSTRALIAN NATIONAL UNIVERSITY  
*First Semester Examination – June 2002*

**COMP4300**  
**Parallel Systems**

*Study Period: 15 minutes*

*Time Allowed: 2 hours*

*Permitted Materials: Calculator*

*Exam questions total 100 marks.*

*Answer ALL parts of ALL questions.*

*Point form is recommended for ‘descriptive’ questions.*

*Clarity and conciseness in answers will be highly valued;  
marks may be lost for supplying irrelevant information in  
answers.*

**Question 1 [2 marks]**

Give a mathematical definition of speedup and efficiency.

**Question 2 [2 marks]**

Give two features that are new to MPI-2 (compared to MPI-1).

**Question 3 [2 marks]**

What is meant by a thread safe routine?

**Question 4 [2 marks]**

What is the significance of the communication/computation ratio?

**Question 5 [4 marks]**

Explain the differences between blocking, non-blocking, one-sided and collective communications?

**Question 6 [4 marks]**

What is Gustafason's law and how does it relate to Amdahl's law?

**Question 7 [4 marks]**

Define two criteria for evaluating static networks. Use these criteria to compare a network based on a hypercube with one based on a 2-dimensional torus with the same number of nodes.

**Question 8 [25 marks]**

- (a) What is load balancing and why is it not practical to find optimal solutions for large load balancing problems?
- (b) The master/slave approach is one method for implementing dynamic load balancing. Outline the master/slave approach and provide pseudo code to illustrate how the method may be implemented in a message passing environment. You can assume that each task is independent, it is determined by a unique identifier and produces a single result that must be communicated back to the master. Tasks do not give rise to new tasks.
- (c) What are the limitations of the master/slave approach and how may they be overcome.
- (d) If the tasks that are being dynamically load balanced give rise to new tasks what complication does this cause? Describe one method for addressing this complication.

## Question 9 [25 marks]

The routine `MPI_Alltoall` sends data from all processes to all other processes and has the following prototype:

```
int MPI_Alltoall( void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 void *recvbuf, int recvcnt, MPI_Datatype recvtype,
                 MPI_Comm comm )
```

You attempted to use this function on the multi-million dollar Kombac 2Me distributed memory parallel supercomputer, but found that it was not working correctly. You have tried contacting the vendor but they have recently merged with Dolet Mestart and are not returning your calls. In desperation you decide to write your own limited implementation of this function using basic send and receive calls. (Fortunately these basic calls are working correctly).

Outline how you would implement a restricted version of `MPI_Alltoall` assuming

- There are a total of  $P$  processes
- Each process sends a message of the SAME length ( $N$  bytes) to all  $P$  processes including itself.
- The routine returns to each process a contiguous buffer of length  $P \times N$  bytes containing the messages sent by every process ordered from process 0 to  $P - 1$ . (Note again that the return buffer includes a copy of its own send message).

You should provide enough information to convey your algorithm and determine that it is functionally correct, i.e. you are NOT expected to write a complete code that is syntactically correct!

Provide a cost analysis of your algorithm assuming that the time required to send a message is given via

$$t_{total} = t_s + m \times t_b$$

where  $t_s$  is the startup latency,  $m$  is the message length in bytes and  $t_b$  is the bandwidth. Thus  $t_{total}$  is independent of how far the message is sent. Note also that the basic architecture of the Kombac 2Me supports multiple simultaneous pairwise communications without contention.

You will be marked on the correctness of your algorithm and its efficiency.

## Question 10 [30 marks]

Discuss shared memory parallel computers from a computer hardware perspective. An incomplete list of issues you should address is given below:

- Different memory layouts and their comparison.
  - What makes a well balanced shared memory computer.
  - Technical limitations in the design of shared memory computers.
  - Cache coherency and why it is important on shared memory machines.
  - Details of at least two commonly used cache coherency protocols.
  - Why memory locks are important in spite of cache coherency.
-