Your Name:

Parallel Programming with Migratable Objects

CS 598 LVK

Exam

Use back sides of paper and extra sheets if necessary.

Problem 1: T/F or Multiple choice:

A. Charm++ interface file must contain nothing but legal C++ code . Circle one: TRUE FALSE

B. A chare that is an element of a chare array remains on the same processor where it started executing, for the duration of program, although its initial placement may be decided by the runtime system anytime before its constructor is called.
Circle one: TRUE FALSE

C. Compared with entry methods with marshaled parameters, methods which take a message as a parameter are slower to use because of the overhead of allocating messages.
Circle one: TRUE FALSE

D. If two prioritized messages (i.e. method invocations) are waiting in the queue on a PE, one with an integer priority of 23 and the other with a priority of 28, the one with priority 23 will execute first.
Circle one: TRUE FALSE

E. When writing a threaded entry method, one has to be concerned about how much memory is being used on the stack, but it is not a significant concern for sdag entry methods
Circle one: TRUE FALSE

F. Charm++ guarantees that two invocations sent between the same pair of chares are delivered in the same order as they are sent, assuming there was no priority associated with either message.
Circle one: TRUE FALSE

G. You can create a one dimensional chare array containing 23 elements, with smallest index being 0 and the largest index is 400.
Circle one: TRUE FALSE

H. The individual iterations of the *forall* construct in structured dagger are meant to be executed on different processors, if other idle processors are available.
Circle one: TRUE FALSE

I. You want a particular computation be carried out on a specific processor. Select which of the following is true: (i) it is impossible to do it in Charm++ (ii) it can be done using chare arrays that are dynamically load balanced, using only one message (iii) it can be done using chare groups, using only one message

J.

Problem 2:

Consider a parallel computation with a one dimensional chare array A of size 200,000 running on 10,000 PEs (processors). So, there are about 20 chares per processor. (for the purpose of this question, assume each PE is a node by itself, and if two chare that are on different processors need to communicate, they need to use the network). The initial mapping of the chares to processors is such that contiguous chares are kept on the same PE. So, chares with indices 0..19 are on PE 0, 20-39 on PE1 and so on.

The computation is iterative; in each iteration, each chare sends its messages, receives message (as described below) and then carries out a local computation. The application runs for 10,000 iterations.

The **communication structure** is such that each chare array element A[i] exchanges messages with chares A[i-1] and A[i+1] (with the leftmost and the rightmost chares communicating just one message). This is a communication intensive application: If the communicating chares are on different processors, they spend t millisecond in communication related functions each iteration, which represents 30% of the computation costs of each chare in each iteration. If the communicating chares are on the same PE, we will assume that the communication cost is zero.

Load Imbalance: the application, when started, is highly imbalanced, with some processors having three times more load than the average. However, the load of individual chare does not change at all across iterations.

Questions:

(A) How the "refinement" and "greedy" load balancing strategies that you used in the programming assignment will compare against each other in this scenario.

(B) Will refinement strategy improve performance compared with no load balancing at all? Explain your answer.

C) Sketch an alternative load balancing strategy that will be better that both *greedy* and *refine.* You don't need to describe its detail, but must include a description of why it will be better.

(C) If we change the problem statement so that: the load changes frequently throughout the execution and also the communication cost is insignificant (say, less than 1% of the computation cost), will that change the answers to parts (A) and (B) above? If so, how? If not, why not?

Problem 3:

Consider the following situation: A chare array B stores a large table of values for indices going from 0 to several billion (i.e. it is a large table which won't fit on one processor). Each chare stores 10,000 consecutive values. So, given an index (also called a *key*) m in the big global table, m/10000 is the index of the chare in B that holds the m'th value. We say B[m/10000] *owns* the key m.

Each element chare of a chare array A needs to do the following, in its entry method F: fill up a local table (i.e. an array) T of size K using data from the table stored in a chare array B. Which values it needs is not known a priori, and K is not known at compile time. It first calculates a set of indices in the big global table that it needs and stores those indices in a local array Q. and then, for each index j in Q, it requests the *entry method G* of the owner of the Q[j] to send back the value corresponding to the key Q[j].

In pseudocode:

Calculate required keys in array Q of size K

For each j between 0 and K, request B[Q[j] / 10000] to send the value corresponding to the key Q[j].

Store the values in a local table T, such that T[i] is the value corresponding to key Q[i].

Obviously, we desire to do this efficiently, so that the calling chare doesn't have to wait for k round trips across the network one *after* the other. This question is about how to express this computation in multiple ways:

(a) can this be be expressed using G as a sync method? Assume F is threaded . Why or why not?

(b) This can be expressed using *futures*, and assuming F is threaded. Describe the pseudocode for F and G for that case. (you don't need to use the actual syntax of future related calls. But to remind you of the functionality, the relevant calls are *CkCreateFuture, CkSendToFuture, and CkWaitFuture*.

(c) This can be expressed using structured dagger notation for F (now, F is not threaded). Describe the pseudocode for F and G for that case. (Hint: you may need to add another entry method H).

Problem 4:

This is a subjective question. It will be graded based on the quality of your reasoning, not which opinions you give.

(A) Describe two issues that make it difficult to develop programs using the Charm++ programming model, and explain why. (This is a question about the programming model. So answers such as "the manual has errors" or "needs more example programs" are not appropriate.)

(B) Describe two features you like about Charm++ as a parallel programming model and explain why.