8.3
The number of nodes is $512/8 = 64$.

(a) 64 bits of directory entry per cache line implies 8 bytes of overhead per 64 bytes of actual memory block, i.e. an overhead of 12.5%. Depending on the protocol, some more bits may be added which will increase the overhead a bit.

(b) We need $\log_2$ (num. of nodes) or 6 bits per pointer. Each memory block has 3 pointers, so the overhead (excluding state bits) is 18 bits per 64 byte memory block or 3.5%.

8.4
Here are the diagrams. L (local) is the node issuing the write. H is the home node of the block. S is a sharer. In general, there may be multiple sharers, though only one is shown here.

Strict Request Reply:

Intervention Forwarding:

Reply Forwarding:
This question is largely answered in the chapter, see Figure 8.13. The main problem other than simplicity is that it is easier to provide fault tolerance in a strict request-reply protocol, since the writer knows the progress of the write operation when the fault occurs.

Here is an example of a good solution:

The first observation, mentioned in the book, is that the head node, when invalidating the rest of its list, sends request/response to each member of the list in order. An obvious optimization is to have each sharer of the list forward the invalidate onto the next sharer, and only the tail responds back to the head. This would reduce the number of invalidation messages - all of which are on the critical path - from 2N to N+1, where N is the number of sharers in the list.

Another possible optimization is when the head node must contact the home node to change the block from FRESH to GONE, and only then invalidate the list. It would be possible to send the invalidations to the sharing list at the same time as the upgrade request is sent to the list, removing the round trip home from the critical path. This optimization is complicated when on the upgrade request to the home node, the head finds that it is no longer the head of the list: another node has beaten the head for an exclusive copy or read-only copy, and the current head hasn't been notified yet. However, I believe that passing the invalidations to the other sharers "early" may not be as big of a problem as it first seems: presumably, the current head who is trying to write will continue to try to obtain the block in exclusive state, and by invalidating sharers early, there are just fewer sharers to invalidate later. Of course, the "current" head would have
to wait for the invalidations to come back before it could remove itself from the list and try to obtain a read only copy again.

A third optimization may be to actually relax that last constraint: remove yourself from the middle of the list (in order to try and become the ONLY-HEAD) at the same time that you invalidate the rest of your list. You can update the state at the home node before you are actually removed from the list, and only when your previous chain is fully invalidated, you inform your previous node that it is now the tail. This scheme would likely involve adding at least one more node pointer to the cache line structures, but is useful both for the problem mentioned above, and when a non-head sharer wants exclusive access.

Optimization 3 (This diagram illustrates the optimization, but is not complete)

In all of these schemes, more state information must be kept, probably in the form of more transient/pending states. But this protocol seems to have a few dozen already, whats a few more?
8.18

4. Text Problem 8.18

a)

Page X: At M₀
  \[ P₀ \text{ Cost (local): } 14 \quad P₀ \text{ Cost (remote): } 56 \]
  \[ p₁ \text{ Cost (remote): } 44 \quad P₁ \text{ Cost (local): } 11 \]
  \[ \text{Total} \quad 58 \quad 67 \]

Page Y:
  \[ P₀ \text{ Cost (local): } 0 \quad P₀ \text{ Cost (remote): } 0 \]
  \[ p₁ \text{ Cost (remote): } 72 \quad P₁ \text{ Cost (local): } 18 \]
  \[ \text{Total} \quad 72 \quad 18 \]

Page Z:
  \[ P₀ \text{ Cost (local): } 15 \quad P₀ \text{ Cost (remote): } 60 \]
  \[ p₁ \text{ Cost (remote): } 37 \quad P₁ \text{ Cost (local): } 9 \]
  \[ \text{Total} \quad 52 \quad 69 \]

Pages X and Z should be placed in M₀, and Page Y should be placed in M₁, because that results in the lowest time for each.

b)

Page X should be replicated, since all accesses are Reads, both get the benefit of local access
Page Y should be migrated to M₁ since P₀ doesn't access it
Page Z should have nothing done since replication is not an option under normal consistency constraints.

c)

Page X should still be replicated. The total cost is:
\[ \text{cost} = \text{replication} + P₀ \text{ Cost (local)} + P₁ \text{ Cost (local)} = 10 + 14 + 11 = 35 \]
which is less than not replicating or moving the page (58).

Page Y should still be migrated. The total cost is:
\[ \text{cost} = \text{migration} + P₀ \text{ Cost (local)} + P₁ \text{ Cost (local)} = 10 + 0 + 18 = 28 \]
which is less than not moving the page (72).

Page Z should still have nothing done, as accessing it from M₀ results in the best time already.

d)

Page X should have nothing done. The total cost of replication would be:
\[ \text{cost} = \text{replication} + P₀ \text{ Cost (local)} + P₁ \text{ Cost (local)} = 60 + 14 + 11 = 85 \]
which is more than the total cost of not replicating or moving the page (58).

Page Y should have nothing done. The total cost of migration would be:
\[ \text{cost} = \text{migration} + P₀ \text{ Cost (local)} + P₁ \text{ Cost (local)} = 60 + 0 + 18 = 78 \]
which is more than the total cost of not moving the page (72).

Page Z should still have nothing done, as accessing it from \( M_0 \) results in the best time already.

\[ e) \]
The costs now become:

<table>
<thead>
<tr>
<th>Page X:</th>
<th>At ( M_0 )</th>
<th>At ( M_1 ) (can't happen here)</th>
<th>Replication:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 ) Cost (local):</td>
<td>140</td>
<td>( P_0 ) Cost (remote): 560</td>
<td>60</td>
</tr>
<tr>
<td>( P_1 ) Cost (remote):</td>
<td>440</td>
<td>( P_1 ) Cost (local): 110</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>580</td>
<td>670</td>
<td>310</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Page Y:</th>
<th>Migration:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 ) Cost (local):</td>
<td>0</td>
</tr>
<tr>
<td>( P_1 ) Cost (remote):</td>
<td>720</td>
</tr>
<tr>
<td>Total</td>
<td>720</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Page Z:</th>
<th>Migration:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 ) Cost (local):</td>
<td>150</td>
</tr>
<tr>
<td>( P_1 ) Cost (remote):</td>
<td>370</td>
</tr>
<tr>
<td>Total</td>
<td>520</td>
</tr>
</tbody>
</table>

Page X should be replicated, for a total cost of 310 as opposed to 580.
Page Y should be migrated for a total cost of 240 as opposed to 720.
Page Z should still have nothing done to it, for a total cost of 520.