Questions about HBFT

Basics
What problem is this paper solving?
What is primary/backup replication?
What is the approach taken here with hypervisors?
What is the I/O accessibility assumption?
- How realistic is it?
What is the Virtual Machine assumption?
- How realistic is it?
Identical instruction streams are important to the approach: why?
What is the difference between ordinary instructions and environment instructions?
- Which is harder to deal with w.r.t. the approach in the paper?
What is the ordinary instruction assumption?
- How realistic is it?
What is the environment instruction assumption?
Why do interrupts have to be delivered carefully?
- What would happen if the system did not manage this?
What is the instruction-stream interrupt assumption?
- How is it realized on PA-RISC?
- How could this be realized with binary translation instead?
Time is thus divided into epochs; when are interrupts delivered?
- How does this impact I/O performance?
When does a backup run an epoch as compared to the primary?
What happens at the end of an epoch?
How are interrupts managed at the backup?
How are environment instructions handled differently on primary/backup?
Does the backup issue I/O instructions in the common case?
What is the two generals problem?
- How does it related to the problem at hand?
How does the system overcome this problem?

Prototype
How do the authors virtualize memory in the hypervisor?
- What restrictions do they impose?
Some instructions check current privilege level
- Why is this a problem?
- How do the authors solve it?
The ordinary instruction assumption does not hold; why?
- What is the more general problem here?
- How is the problem solved? How general is this solution?

Performance
What is the main cause of overhead in the system?
What parameters affect it?
What overhead is actually measured (and shown) in Figure 2?
Does I/O performance strictly improve with longer epochs?
What experiments didn't the authors perform that you would like to see?

Reprise
What problem is this paper solving?
How effectively is it solved?
When would this be useful?
What are other ways to build fault tolerant services?
How do they compare to what is done in this paper?

Random
How much code did they write?
Best line: "A solution (hack) was to modify this code fragment…"