Topic: Hyper-visor based fault tolerance

What is the paper about?
This paper presents a method for building reliable fault-tolerant computing systems by replicating machine state using a hypervisor. The system will tolerate faults in the CPU or memory caused by power issues, overheating, or full destruction of the primary system (like in military applications). The approach presented does nothing for disk failures or software faults, however.

What is the IO accessibility assumption?
IO Accessibility - this paper assumes all IO devices are accessible from both the primary and backup systems- as is the case when a disk is placed on a shared SCSI bus.

Why are identical instruction streams are important ?
The approach presented ensures that both the primary and backup systems execute identical instruction streams - complete with identical interrupt delivery at the same point in each instruction stream. This ensures that the state of both the primary and all backups remain identical with no attempt to 'checkpoint' or verify that the state is the same between multiple systems.

Why are interrupts a problem?
When interrupts come in i.e the timing of their arrival might cause a different set of instructions to be executed.
How is this deal with?
Batching is followed here. Although the benefits of coalescing are not got here.

What is the difference between Ordinary instruction and environment instructions?
The paper distinguishes between ordinary and environment instructions. Ordinary instructions (add; ld; st) are guaranteed to execute identically on both platforms if the state is originally the same. Environment instructions (getttimeofday) are influenced by external factors. To ensure identical execution, the hypervisor must emulate these instructions on the

When a backup issues a write, how is it managed?
It is ignored. When it does things that is visible to others , like updates , it is ignored.
system. This is done by feeding the answer received from the primary system's execution of any environment instruction to the backup system, and the backup system's hypervisor delivers this response to the backup system.

When does back up run an epoch compared to primary?
Because of this synchronization, the backup system always runs 1 epoch behind the primary system since it requires 'answers' to be provided from primary to backup.

How realistic is the Ordinary instruction assumption ?
The Ordinary instruction assumption made by them is not very realistic since different contents in the TLBs of the processors would result in some extra instructions(TLB miss handler) running on some processor due to TLB misses and not the others. Thus the “same” set of instructions are not run on all processors.

TLB miss handling on the HP PA-RISC computers is non-deterministic- which posed a challenge for this implementation. To guarantee deterministic synchronization between the primary and backup, TLB miss handling was handled by the hypervisor to ensure that it would be deterministic. Because of this, OS TLB miss handling routines are never called giving the appearance to any OS that hardware is managing the TLB.

Just as environment instruction 'answers' are back-channeled from primary hypervisor to backup hypervisor, the same is done with interrupts. The goal is to insure that the backup executes an identical instruction stream, so all 'normal' interrupts are discarded on the backup system by the hypervisor, instead, the backup hypervisor receives a stream of interrupts from the primary via the network, and 'replays' these to the backup guest OS at the same point in the instruction stream as the primary and in the same order. Any 'outgoing' IO requests (like writes to disk) are thrown away by the backup hypervisor- allowing only the primary system to make changes as long as it's active.

Is IO performance better or worse because of epoch?
It is important to note that I/O performance is worse because of epochs as they just add latency without any benefits of coalescing.

When failover occurs (backup takes over for a failed primary system), it's possible that many output instructions (like writes to disk) will be repeated. The backup may not know precisely when the primary failed (or why) and the primary runs an entire epoch ahead of the backup. Instead, output instructions are simply repeated. The authors argue that IO devices are used to

these kind of shenanigans (repeating output instructions) and deal with them just fine. It's worth noting that they're probably right.

This hypervisor was designed with no extra level of memory virtualization allowing them to sidestep the thorny problem that all later virtualization designs must contend with. Instead, the hypervisor exists in the guest OS's address space and 'appears as another device driver'.