



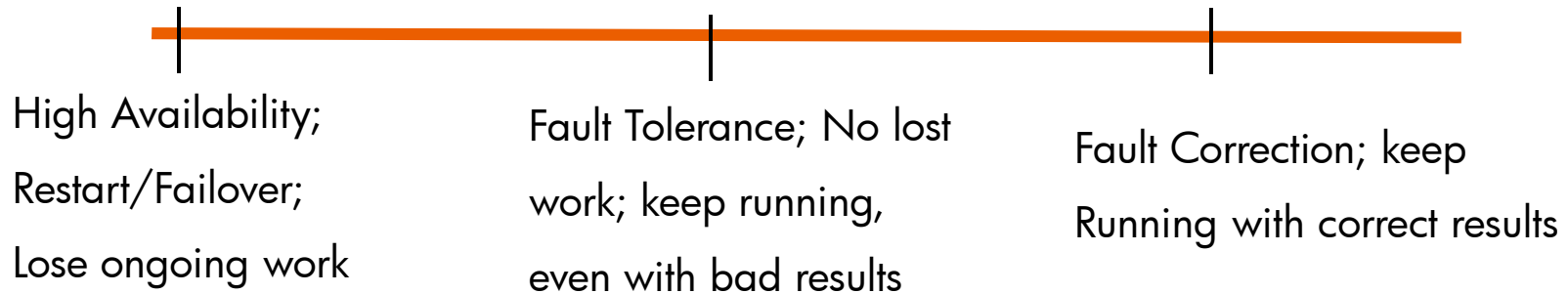
# Detecting and Correcting Transient Hardware Errors

**John Byrne (john.i.byrne@hp.com), Norman P. Jouppi,  
Laura Ramirez, Parthasarathy Ranganathan, Bruce J. Walker  
HP Labs**

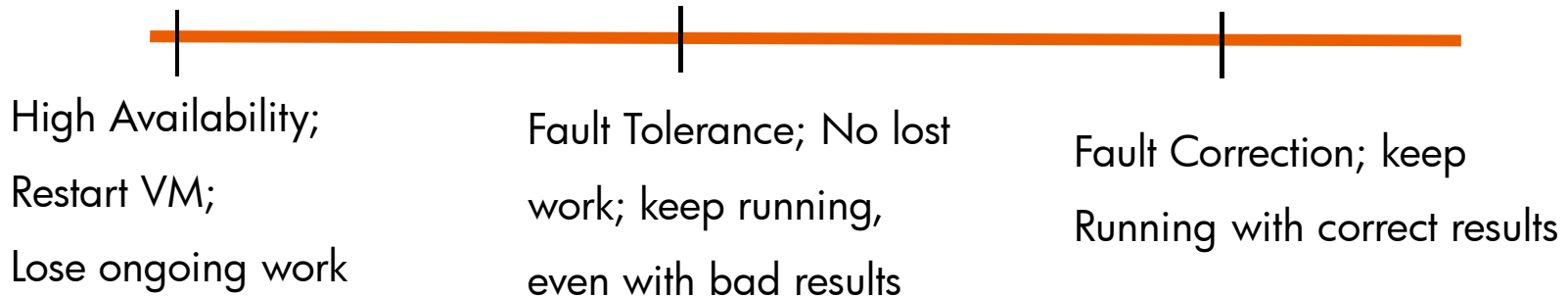
**Nidhi Aggarwal, Kewal K. Saluja, James E. Smith  
University of Wisconsin – Madison**







# Availability/Reliability Spectrum



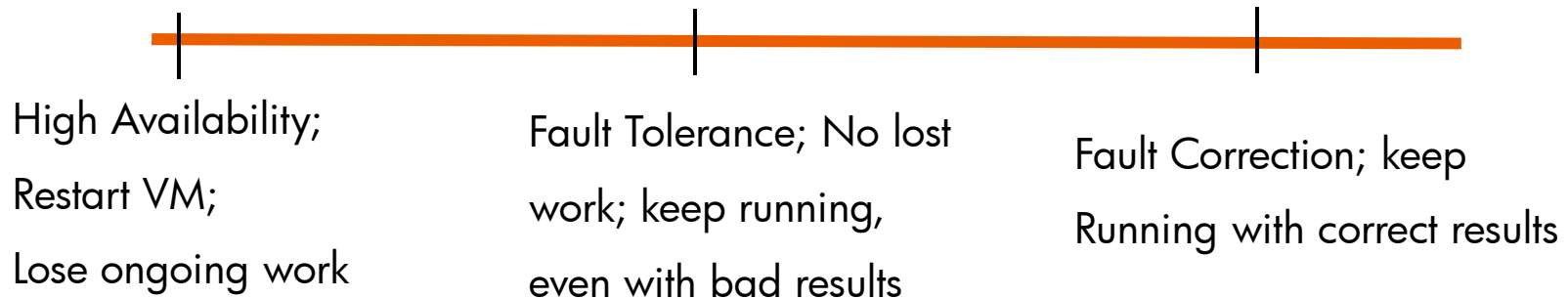
# Availability/Reliability Spectrum







## Ongoing FT Work:

	Remus		Checkpoint and restart on complete node failure
	Kemari		
Lockstep	Marathon		Output comparison; pick one if they don't match
Execution	VMware		Backup takes over on complete node failure

# Availability/Reliability Spectrum



## Ongoing FT Work:

	Remus		Checkpoint and restart on complete node failure
	Kemari		
Lockstep Execution	Marathon		Output comparison; pick one if they don't match
	VMware		Backup takes over on complete node failure

Theme: Deal with Complete Node Failure

No one is detecting or correcting transient processor failures

# Transient Hardware Errors

- International Technology Roadmap for Semiconductors has predicted significant reliability problems
- Intel study in 2005 indicated 100-fold increase in transient faults in scaling from 180nm to 16nm

Errors can:

- a. Crash the OS;
- b. Corrupt data;
- c. Cause execution to take a different path;

Goal: Detect and Correct Transient Errors

# Detect and Correct Transient Errors

1. Lockstep VMs with ongoing checkpoints.
2. Tee input to both VMs
3. Compare output from VMs and re-execute on miscompare
4. Compare checkpoints and re-execute on miscompare
5. Log input, interrupts and non-deterministic instructions to allow completely accurate re-execution;

# Lockstep VMs

- Create 2 identical images at VM start time;
- Ensure response to each VMexit is identical;
  - Force them to be identical if necessary (rdtsc)
- Deliver interrupt at the identical instruction
  - At each VMexit, deliver interrupts if they are pending;
  - Use the count-down PMU counter to force a synchronization point if no VMexits happen and deliver interrupts at that point.
- Log VMexit return values as necessary (for replay);
- Log when interrupts are delivered (for replay)

# I/O

- Input is sent to both VMs (network and disk);
  - Input is logged as being part of a specific checkpoint so on replay inputs can come from the log;
- Output is compared and if equal, is sent out;
  - If not equal, re-execute from the last good checkpoint
  - Blktap driver modified to allow disk output to be compared
  - Network backend driver modified to allow network output to be compared;
  - Output is counted so on replay the correct number of re-created outputs can be discarded and not output twice.

# Checkpointing and Comparing

- Incremental, periodic checkpoints of each VM
  - At exactly the same instruction;
  - Utilize COW or copy at checkpoint time;
  - Mark the checkpoint event in the input and output streams
  - Immediate continue execution after checkpoint done;
- After checkpoint X is done, compare the incremental checkpoints
  - If equal, delete one of the checkpoint
  - If equal, delete checkpoint X-1 + any logs for x-1;
  - If not equal, then do a replay from checkpoint X-1;
- At checkpoint event, tell input to start a new log;
- At checkpoint event, tell output to record the output count and start a new count for the next checkpoint

# Replay from Checkpoint X

- Restore the registers and memory image
- Tell input i/o to replay input from log for checkpoint X
- Tell output that we are replaying from checkpoint X and it then throws away as many i/o's as were done since checkpoint X.

# Initial Limitations

- Uniprocessor VMs
- HVM guests
- Single Node implementation

# Performance Data to Date

- Implementation to date includes lockstep VMs and disk i/o input funneling and output checking (network i/o is implemented but not yet tested)
- Bonnie i/o benchmark didn't show any degradation;
- SpecCPU benchmark suite showed 2-5% degradation.

# Plans

- Implement the checkpoint and replay code
- Work on performance of lockstep and checkpoint
- Investigate UP, HVM and single-site restrictions.

**LABS<sup>hp</sup>**

