



We Make Software Behave

Reliable | Software
Technologies

Experiences with OS Reliability Testing on the Exemplar System

How we built the CHO test from recycled materials



We Make Software Behave

CHO version 1

0:00	shell_stress (SPECint cont'd)		
0:30			app
1:00			vmstress
1:30			
2:00			
2:30			
3:00			app
3:30			
4:00			
4:30			
5:00	SPECint		
5:30			

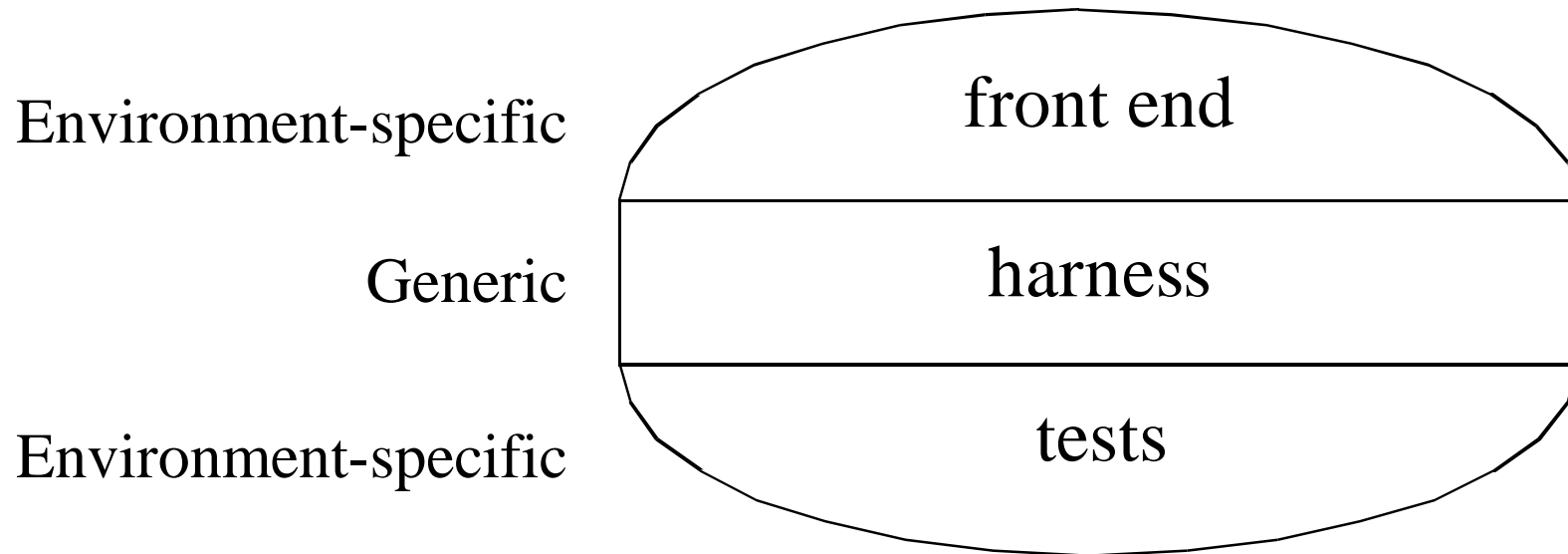


Sweat sequence file

Test	Timeout, How	Delay
shell_stress	0,pass	0
app	360,pass	120
app	420,pass	720
vm	720,pass	240
SPECint	900,pass	1200



The Sandwich Model





Scaling

- **The user application and SPECint did not scale.**

- **vmstress-**

$$\text{(free_memory_pages - 1000) * (stress_factor + 3) / 10}$$

- **shell_stress-**

$$\text{max} = \text{cpus} * (\text{stress_factor} + 3)$$

$$\text{min} = \text{max} * .75$$



We Make Software Behave

CHO Version 2

0:00	shell_stress	gsmstress	Gaussian92
0:30		pong	
1:00			
1:30			Naspar
2:00			
2:30			
3:00		ftpstress	
3:30		pong	
4:00			
4:30			Gaussian92
5:00			
5:30			



Sample run, basically healthy

Machine: bop-f (4 nodes, 32 total cpus)

CHO Version: 2.0.6 OS 96.08.05.00

Duration: 19.53 hours Stress_factor: 1

Peak Users: 65 (range 48-64 + console)

Peak Load Average: 65.14

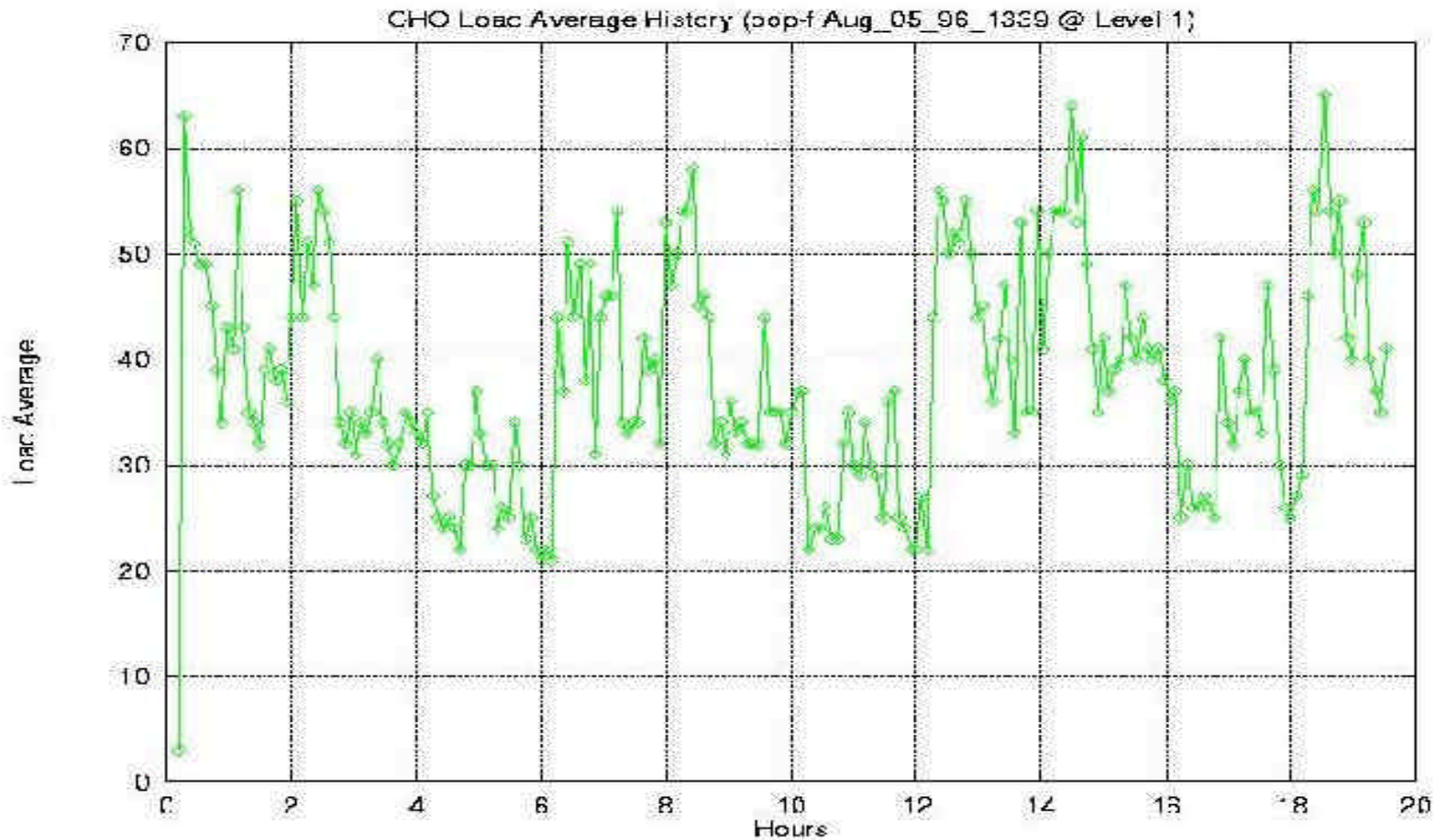
Failure summary: MU on node 0 CPU 5 became unreachable. No lcds.

The next two slides show a graph of the load average and user count over time. Load average shows three cycles through the sequence file. User count shows random variation in shell_stress.



We Make Software Behave

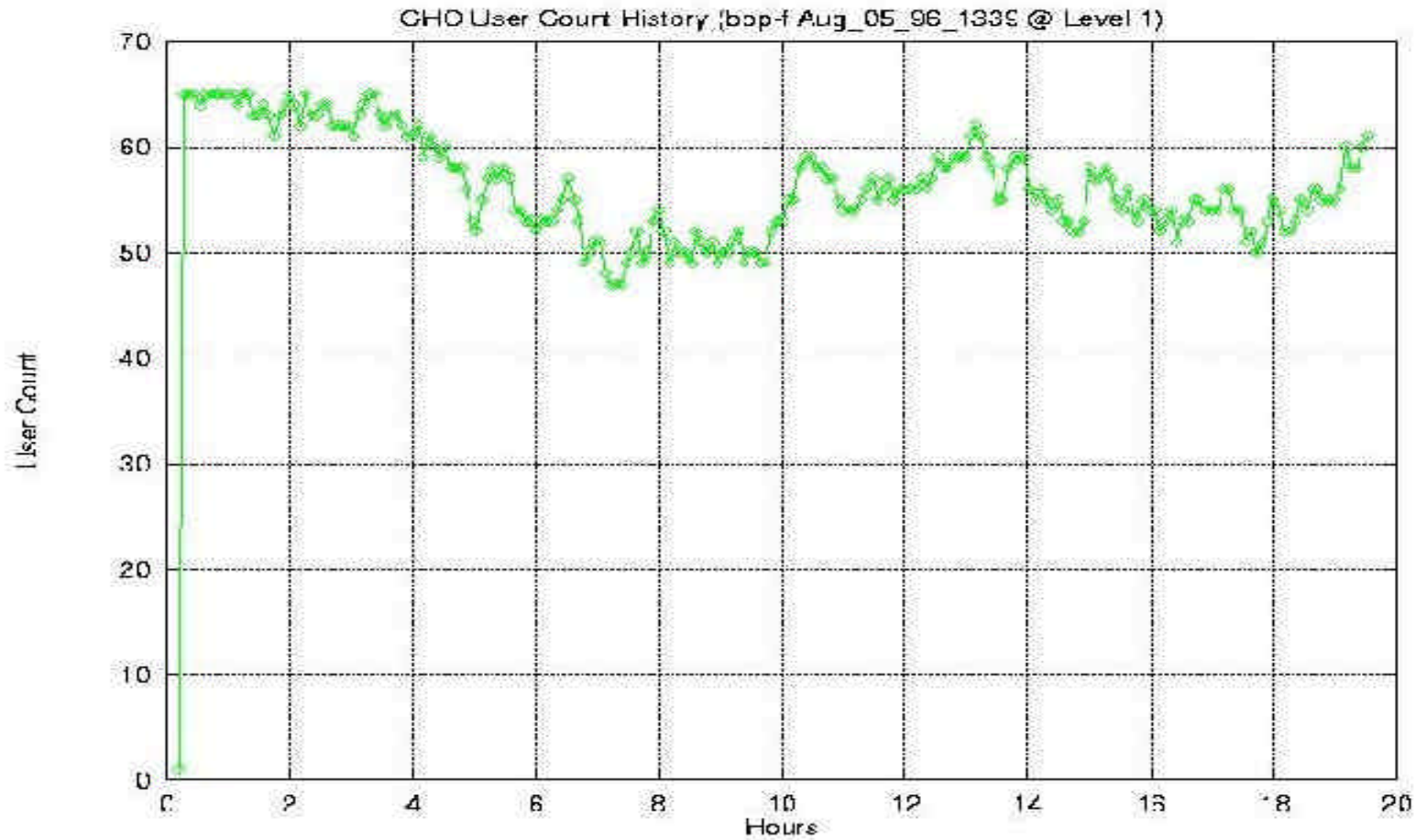
bop load average





We Make Software Behave

bop user count





Sample run, unhealthy

Machine: hunter-f (4 nodes, 32 cpus)

CHO Version: 2.0.6 OS 96.08.14.02

Duration: 31.28 hours Stress_factor: 10

Peak Users: 157 (range 120-160 + console)

Peak Load Average: 40.43

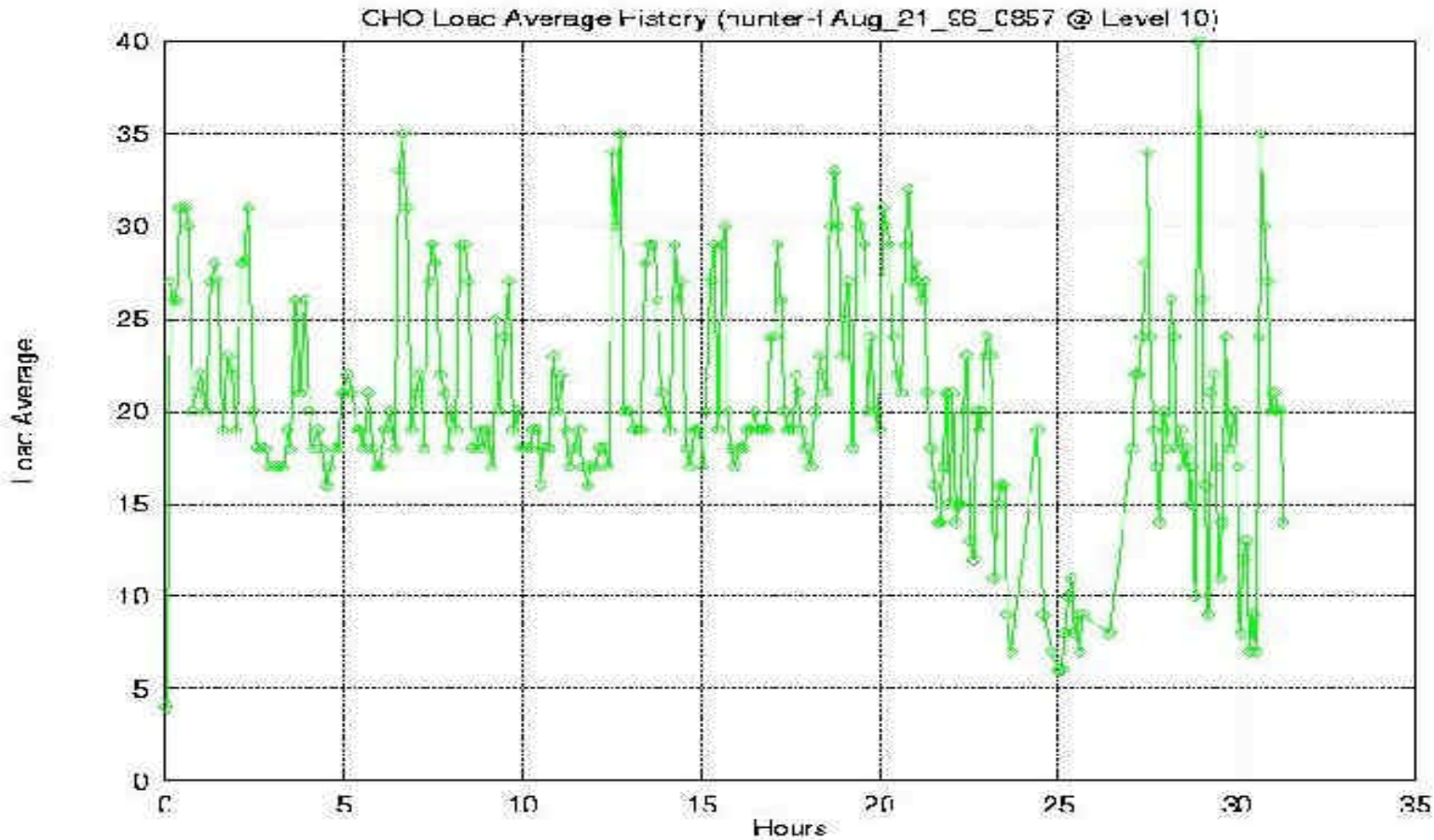
Failure summary: Node 2 had 120 remote page faults outstanding.
Node 1 had 27.

The load average graph shows something sinister at the 21 hour mark, which the user graph confirms. However, the user graph also shows a steady decline from the beginning, which means any uptime number reported may be invalid.



We Make Software Behave

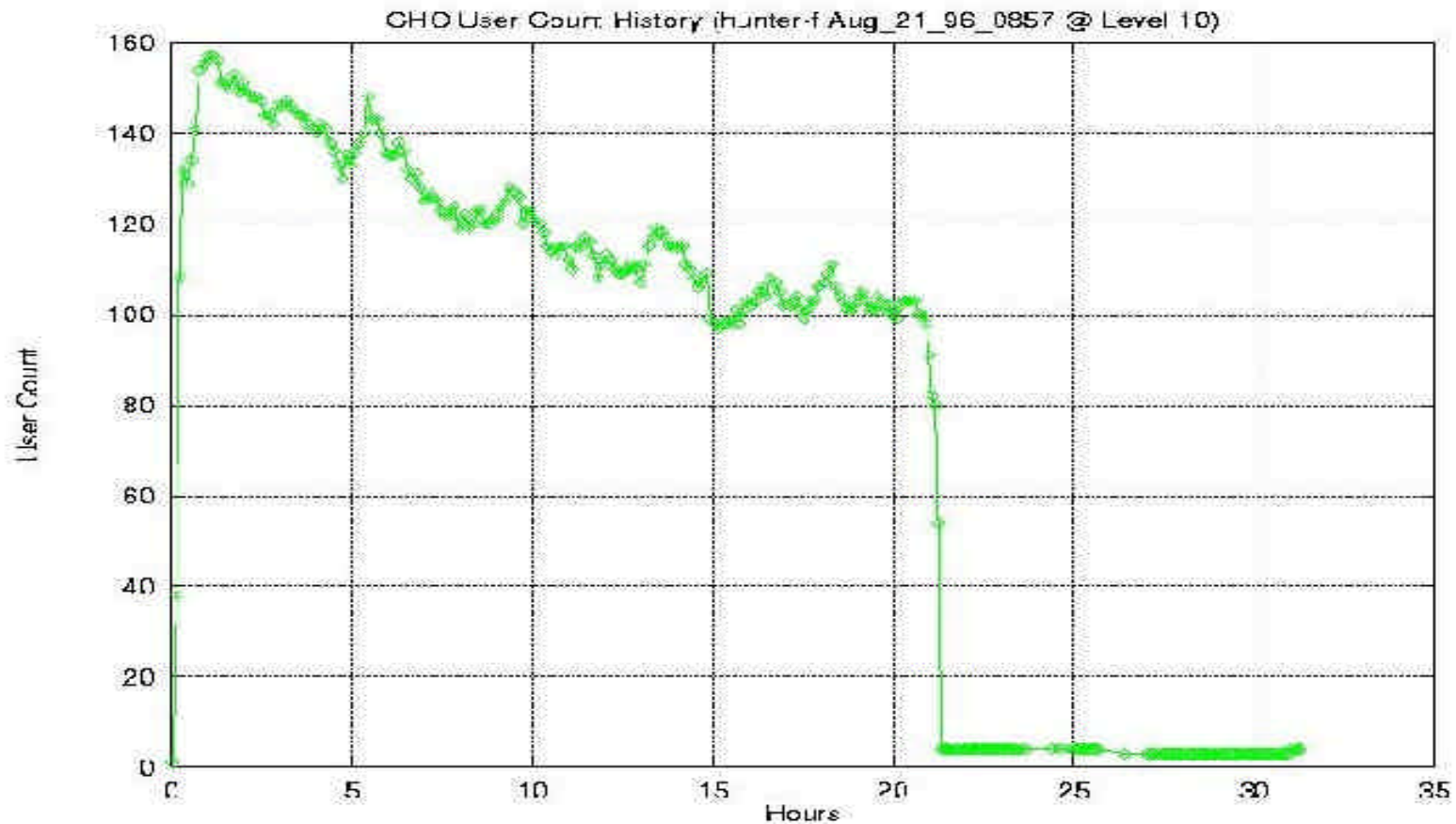
hunter load average





We Make Software Behave

hunter user count





Bathtub curve for software?

A single run of a software system seems to resemble the bathtub curve of failure rates for hardware.

Soon after starting, you hit the initialization bugs.

Parts like virtual memory and stray pointers hit after a long run time.

