

# VM Performance History

SEAS

7Oct1986

Lynn Wheeler  
lynn@garlic.com

360/67

- 750 ns memory
- Simplex - 256k-1024k
- Duplex – 512k-2048k
- Address Translation
  - 24bit
  - 32bit

# CP/67 Release 1

- 10 level dispatcher
  - Based on cpu use
  - High processor overhead
- No real storage throttle

# CP/67 Release 2

- Two level dispatcher (Q1 & Q2)
  - Essentially round-robin
  - Every pass thru dispatcher look at all tasks multiple times
    - 30-40 users, 10% cpu overhead
- Thrashing throttle based on fixed table
- Kernel storage management significant
  - Upwards 1/3<sup>rd</sup> CP overhead

# Work as undergraduate

- Fastpath
  - Kernel SVC linkage reduced 185ms to 65ms
  - Virtual SVC reflection 25ms eliminating dispatch
  - Program interrupt handler
  - Dispatcher
- BALR linkages
- Dynamic adaptive controls
- Local LRU & Working Set thrashing throttle
- Pagable kernel
- Custom I/O for CMS file i/o
- Ordered seek & chained requests

# 68 Share Presentation

- OS/360 MFT14
  - Base 322secs elapsed time
- Base & Modified CP/67
  - Reduced CP CPU from 533secs to 113secs

	W/CP	ratio	CP CPU
original	855	2.2	533
changed	435	1.35	113

# CP67 Release 3

- FastPath
  - Fast redispatch
  - Fast SVC reflect
  - Fast instruction simulation restart
- Restructured Dispatcher
  - Separate in-q chain
- Kernel storage subpool
- CMS Diagnose I/O

# CP/67 Release 3.1

- Dynamic Adaptive Scheduler
- Working set page thrash
- Global LRU Replacement
- Limited avail. Support for V=R



# CP67 Release 3.2

- Internal 3.1L
  - Development group focused on vm370
- Ordered seek queuing
- PCI interrupts
- Chained requests
  - Increased 2301 from 80/sec to 300/sec

# VM370 Release 1

- Lots of Simplification
- No Fastpath, dynamic adaptive, etc
  - Q1 always ahead of Q2
  - Quanta limit only virtual CPU
  - One case Q1 ran for 20mins
- Shared segments
- Local (not global) LRU
- Approx. round-robin

# VM370 Release 2

- Some fastpath (release1 PLC9)
- VMA Hardware support
- CPU use based on both virtual and CP
- VM/VS1 handshaking

# CSC/VM (release 2)

- Relocatable Shared Segments
- Paging Access Method
- CP67 Dynamic Adaptive
- CP67 working set & global LRU
- CP67 Fastpath
- Restructure page supervisor
- Page & Swappable migration
- Q3
- Autolog

# SHARE VM Scheduler White Paper

- VM370 is regression from the best of CP67
- Proposed
  - Additional I/O measurements
  - Resource targets include I/O
  - Runlist limit include more than working set
  - Group scheduling

# VM370 Release 3

- VMA support for VM with shared segments
- DCSS subset of CSC/VM (but w/o PAM)
- AUTOLOG command

# Resource Manager (VM370 3.4)

- CP67 Dynamic Adaptive
- CP67 working set and global lru
- CP67 fastpath
- Restructure page supervisor
- Page & swappable migration
- Q3
- Reliability and cleanup of over 60 modules

# VM370 Release 3.8 ECPS

- Enhanced VMA function
- Virtual timer support
- E6 opcodes
  - Kernel code moved to microcode for 10:1 improvement
  - Top 6k bytes of code, approx.80% of CP cpu



# VM370 Release 4 SMP

- Dependent on lots of code in resource manager
- Resource manager was 1<sup>st</sup> priced SCP
- Free SMP code couldn't require priced code
- Moved possibly 80% of code into free base w/o changing RM price

# SJR/VM Release 5

- Block Pre-paging (track previous pages)
- SYSPAG
- Single chain for eligible list
- Simple group scheduling
- Restructure IOS for performance & availability
- CMS chained terminal I/O
- Restructure CMS sysgen
  - Multiple file directories in shared segments

# SJR/VM Release 5.12

- Extensive timing measurements
  - Runnable & non-runnable measured
  - Restructured Q3 controls
  - IOBLOK queued and service times
  - PAGE I/O queued and service times

# VM370 Release 6 (SEPP)

- CMS EDF
- Additional ECPS support for CMS
- S&Y directories in shared segment
- Uniprocessor V=R support

# PAM CDF & EDF

- Typical application do some file operation
  - 4k/EDF I/Os for physical I/Os, PAM I/O are no. 4k page transfers

	CPU	I/O	elapsed
4k/EDF	3.72	1958	82
PAM/EDF	3.41	3836	56
PAM/CDF	3.02	3790	52

# VM/SP-HPO Release 1

- CP does block 3270 for multiple simulated line
- SMP reworked for single, non-SMP guest
- Support for hardware “protect”
- Significant increase in CP overhead masked by other changes

# CP67-3.2 v. HPO2.5

machine	360/67	3081	ratio
MIPS			40-50
names	105	7000	66
users	80	320	4
channels	6	24	4
drums	12m	72m	*6
Dane i/o	150	600	4

# VM/SP-HPO Release 3.x

- Cache sensitive kernel storage for SMP
- “true” runlist
- 300 mills. Queue drop delay
- Big pages
- SYSPAG



# Later VM/HPO

- Global LRU
- Numerous fixes for “enhancements” from 2.5-3.4 period

# VM Migration Aid

- Originally internal use only for MVS/XA development
  - Originally VM370 killed and all people moved to support MVS/XA
  - Endicott managed to save VM370 product mission but had to recreate group from scratch

# Cluster z/VM

- Minor mention from Hillgang 2009 presentation
- From the Annals of Release No Software Before Its Time
- Internal US HONE datacenters (world-wide sales & marketing support) consolidated in silicon valley in mid-70s.
- Implemented load-balancing and fall-over across large loosely-coupled cluster with large shared disk farm.

# VM Performance History

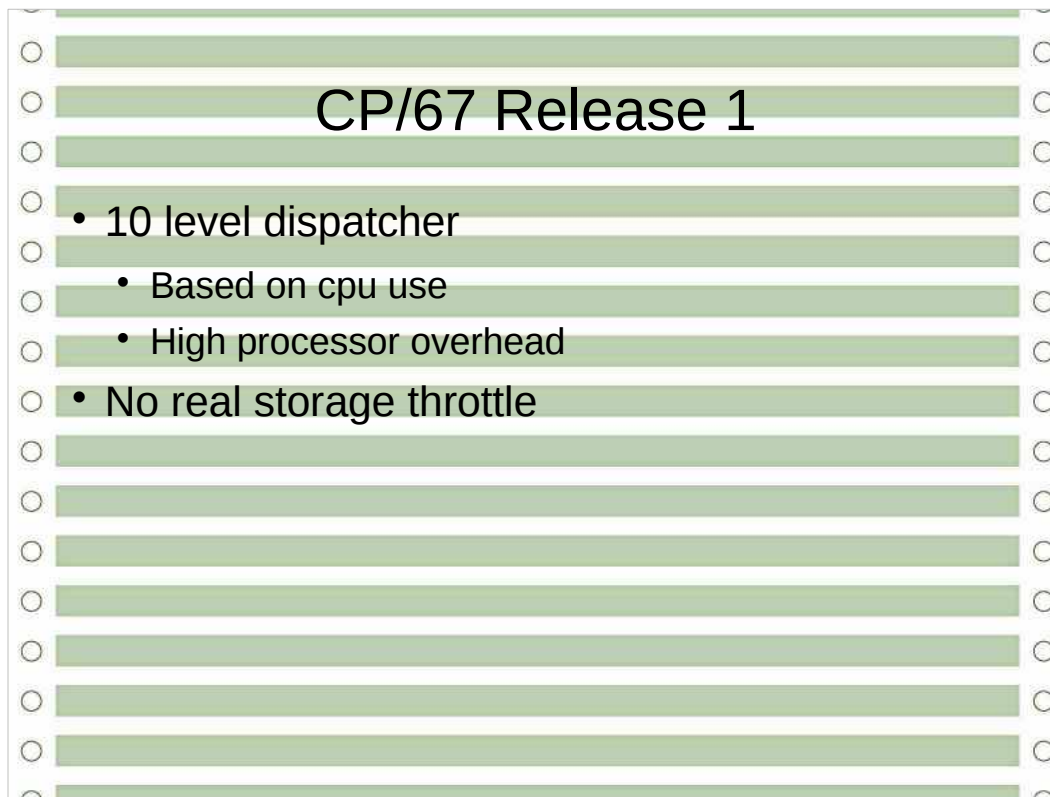
SEAS  
7Oct1986

Lynn Wheeler  
lynn@garlic.com

360/67

- 750 ns memory
- Simplex - 256k-1024k
- Duplex – 512k-2048k
- Address Translation
  - 24bit
  - 32bit

Cambridge has been working on virtual operating system (cp40) using custom modified 360/40 before 360/67 became available.



The first release of CP/67 still bore quite a bit of a CTSS flavor from the stand-point of performance and scheduling. There were no page thrashing controls (CTSS had been a roll-in/roll-out swapping system). The allocation of resources in some cases could consume more resources than the resulting resources allocated.

CP/67 Release 2	
• Two level dispatcher (Q1 & Q2)	
• Essentially round-robin	
• Every pass thru dispatcher look at all tasks multiple times	
• 30-40 users, 10% cpu overhead	
• Thrashing throttle based on fixed table	
• Kernel storage management significant	
• Upwards 1/3 <sup>rd</sup> CP overhead	

The next release of CP/67 had a drastically simplified dispatcher (from ten levels to two), and included an eligible queue for limiting page thrashing, done by Lincoln Labs. The page thrashing controls consisted of limiting the number of in-queue Q1 and Q2 virtual machines to absolute numbers. The numbers were set based on the number of available 256k real storage boxes. The selected values were established by Lincoln Labs. based on the execution characteristics of their local users

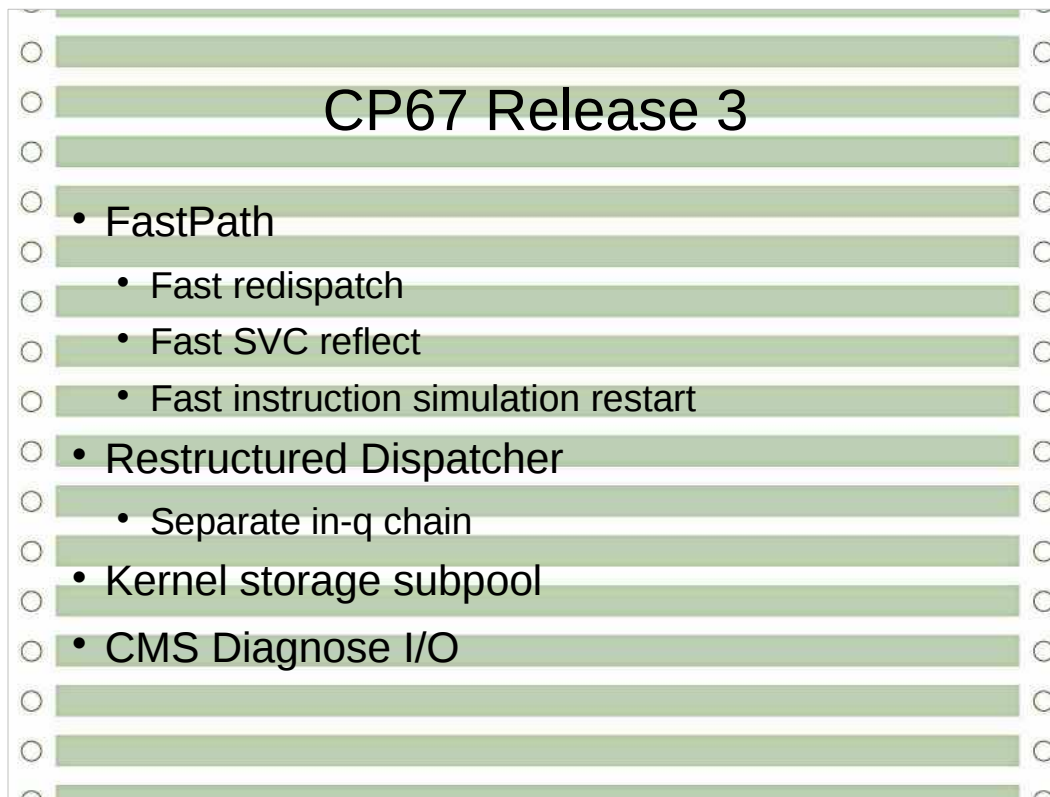
The dispatching and scheduling controls were still very simple and straight forward. CPU overhead still tended to grow non-linear (events\*users). There were no specialized data structures and/or implicit assumptions about possible state changes. Every time the dispatcher was entered, it would check all logged on virtual machines for state changes (exp: check for reflecting virtual I/O interrupts). The chain of UTABLEs was then scanned a second time, searching for any stacked CPEXBLOKs. Finally, the UTABLE chain was scanned a third time to find the virtual machine with the best dispatching priority.





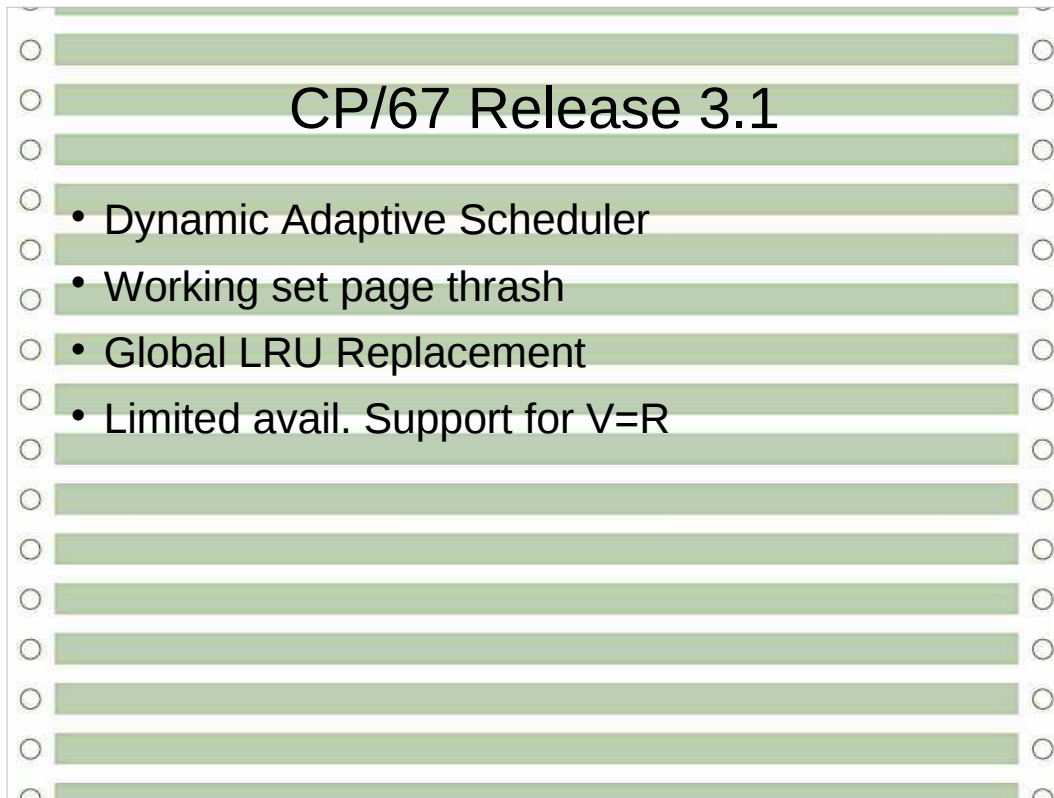
68 Share Presentation			
• OS/360 MFT14			
• Base 322secs elapsed time			
• Base & Modified CP/67			
• Reduced CP CPU from 533secs to 113secs			
	W/CP	ratio	CP CPU
original	855	2.2	533
changed	435	1.35	113

Bare Machine times were 322 seconds elapsed (12.9 sec/job). Time to run just JCL for jobs, 292 seconds (11.7 sec/job). Same run in unmodified CP/67 took 855 seconds to run (34.2 sec/job), with no other workload. Time to make just JCL run was 787 seconds (31.6 sec/job). An attempt was also made to show that the stand-alone to CP/67 runtime ratios were dependent upon the percentage of privileged instructions executed, with the O/S job scheduler having a particular high percentage.



Release 3 of CP/67 had several significant performance improvements, primarily involving the reduction of CPU overhead. Several people at Cambridge were responsible for this effort. Some of the items were direct copies of activity that I did prior to joining IBM. CMS diagnose I/O was a direct descendent of CMS stylized I/O, but a completely different implementation.

FREE subpool support was a completely different approach to free storage management. The traditional pathlength optimization had just about been carried as far as it could go in optimizing the existing implementation. The subpool design recognized that the majority of the CP FREE storage requests were of a specific nature. The subpool logic created a specialized data structure and software support that were specifically tailored to those types of requests (i.e. another flavor of fastpath). This change reduced the FREE storage CPU management overhead by better than a factor of 10 (which had been running better than 20 percent).



The image shows a table of contents for CP/67 Release 3.1. The table consists of 15 rows, each with a small circle on the left and right sides. The first row is empty. The second row contains the title 'CP/67 Release 3.1'. The third row is empty. The fourth row contains a bulleted list item: '• Dynamic Adaptive Scheduler'. The fifth row contains a bulleted list item: '• Working set page thrash'. The sixth row contains a bulleted list item: '• Global LRU Replacement'. The seventh row contains a bulleted list item: '• Limited avail. Support for V=R'. The remaining eight rows are empty.

CP/67 Release 3.1
• Dynamic Adaptive Scheduler
• Working set page thrash
• Global LRU Replacement
• Limited avail. Support for V=R

While release 3.0 mostly implemented changes that directly reduced CPU overhead, release 3.1 implemented more complex algorithm changes (some of which also had the side-effect of reducing CPU overhead), attempting to dynamically adapt the system operation to both a large variety of different configurations as well as load variability either in time (i.e in the same installation at different times of the day) or space (i.e. at different installations).

	CP67 Release 3.2
	• Internal 3.1L
	• Development group focused on vm370
	• Ordered seek queuing
	• PCI interrupts
	• Chained requests
	• Increased 2301 from 80/sec to 300/sec

The development group had split from the science center and were concentrating on VM370. CSC continued on a CP/67 base. With the pre-occupation with vm370, it was decided to package some amount of the CSC work for 3.2

○		○
○	VM370 Release 1	○
○		○
○	• Lots of Simplification	○
○	• No Fastpath, dynamic adaptive, etc	○
○	• Q1 always ahead of Q2	○
○	• Quanta limit only virtual CPU	○
○	• One case Q1 ran for 20mins	○
○	• Shared segments	○
○	• Local (not global) LRU	○
○	• Approx. round-robin	○
○		○
○		○
○		○

CP67 morph into VM370 cleaned up, reorganized and simplified much of the code. CMS reorganized to use 370 64kbyte shared segments and 370 hardware segment protect. Hardware segment protect was dropped as part of 165 hardware schedule problem and the crude storage protect key hack had to be substituted.

○		○
○	VM370 Release 2	○
○		○
○	• Some fastpath (release1 PLC9)	○
○	• VMA Hardware support	○
○	• CPU use based on both virtual and CP	○
○	• VM/VS1 handshaking	○
○		○
○		○
○		○
○		○
○		○
○		○
○		○
○		○
○		○

I contributed some fastpath changes for PLC9.

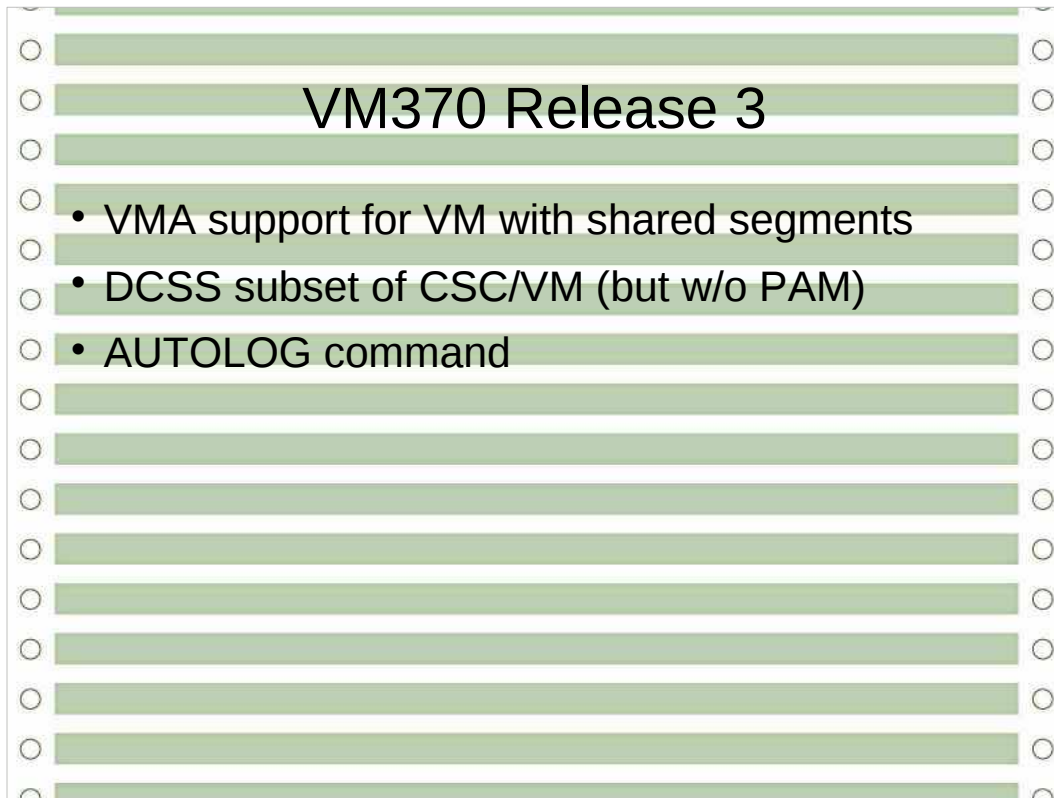
	CSC/VM (release 2)
	• Relocatable Shared Segments
	• Paging Access Method
	• CP67 Dynamic Adaptive
	• CP67 working set & global LRU
	• CP67 Fastpath
	• Restructure page supervisor
	• Page & Swappable migration
	• Q3
	• Autolog

.CSC finally got around to replacing their 360/67 with a two megabyte 155. As part of the conversion, I upgraded much of the CP/67 work to VM/370 (although some stuff got dropped, for instance self-stealing and some other page replacement algorithm bells and whistles). Also, during the conversion period there were a number of enhancements added I created the AUTOLOG command as a part of a set of procedures for automated, unattended benchmarking. Some number of workloads were extreme stress that required lot of vm370 cleanup. Final set for release of resource manager involved 2000 benchmarks that took 3months elapsed time.

## SHARE VM Scheduler White Paper

- VM370 is regression from the best of CP67
- Proposed
  - Additional I/O measurements
  - Resource targets include I/O
  - Runlist limit include more than working set
  - Group scheduling





The image shows a table of contents for 'VM370 Release 3'. The table consists of 16 rows, each with a small circle on the left and right sides. The first row is empty. The second row contains the title 'VM370 Release 3'. The third row is empty. The fourth row contains a bullet point: '• VMA support for VM with shared segments'. The fifth row contains a bullet point: '• DCSS subset of CSC/VM (but w/o PAM)'. The sixth row contains a bullet point: '• AUTOLOG command'. The remaining seven rows (rows 7 through 13) are empty. The last two rows (rows 14 and 15) are also empty.

	VM370 Release 3
	• VMA support for VM with shared segments
	• DCSS subset of CSC/VM (but w/o PAM)
	• AUTOLOG command

VMA for shared CMS was justified on checking 16 shared pages on every task switch being less than benefit gained from VMA. However before release 3 shipped, subset of the CSC/VM CMS changes, doubled the number of shared pages (and shared page checking overhead).

○		○
○	Resource Manager (VM370 3.4)	○
○		○
○	• CP67 Dynamic Adaptive	○
○	• CP67 working set and global lru	○
○	• CP67 fastpath	○
○	• Restructure page supervisor	○
○	• Page & swappable migration	○
○	• Q3	○
○	• Reliability and cleanup of over 60 modules	○
○		○
○		○
○		○
○		○

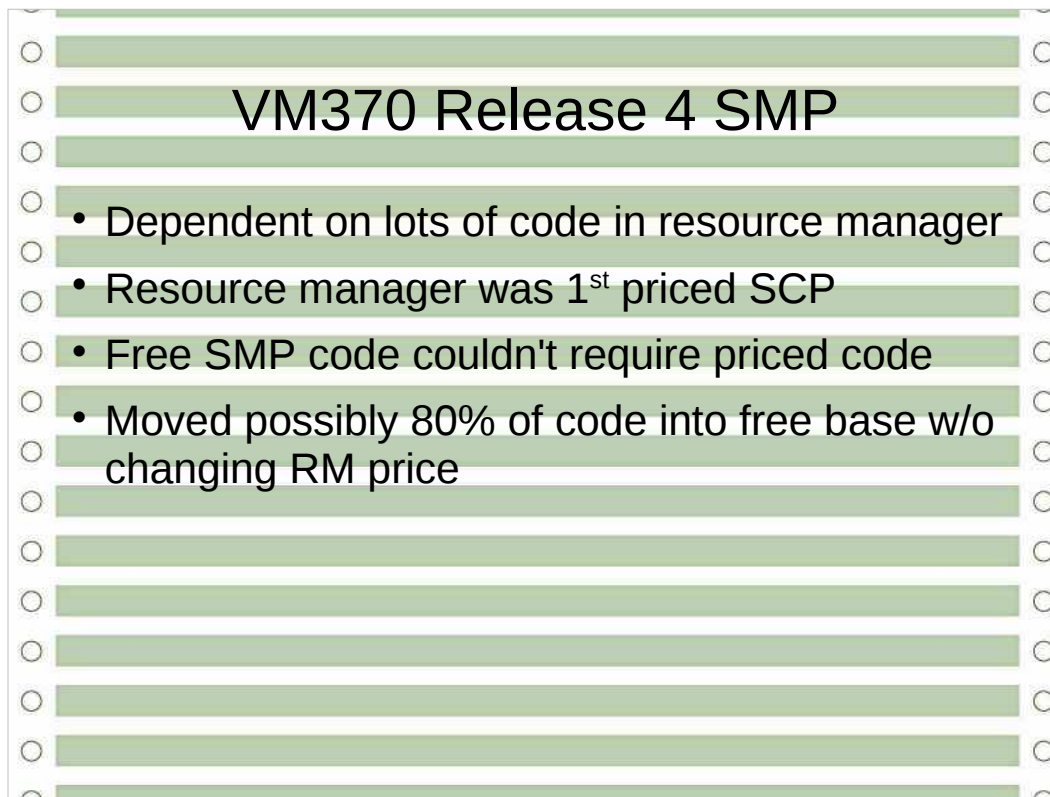
The Resource Manager was announced for release on May 11, 1976. The changes for the PRPQ were extracted from the IBM internally distributed CSC/VM. The announcement letter lists the following highlights: 1) Scheduling Algorithm, 2) Page Migration, 3) Swappable Migration, 4) Reset pages and time stamp segments, 5) working set estimate, 6) fast redispatch extension, 7) enable window, 8) set favored extension, 9) indicate command extension, and 10) selective path length reductions.

There were final 2000 benchmarks that took 3 months elapsed time for RM release.

Resource Manager was selected to be guinea pig for charging for kernel software

	VM370 Release 3.8 ECPS
	• Enhanced VMA function
	• Virtual timer support
	• E6 opcodes
	• Kernel code moved to microcode for 10:1 improvement
	• Top 6k bytes of code, approx.80% of CP cpu

I had started work in Jan of 1975 on a project involving an experimental (5-way SMP) machine and the placement of significant portions of the CP supervisor into the microcode in an architected way. In May, several people from the Endicott visited CSC to discuss moving subset of CP kernel code directly into microcode



SMP design was dependent on a large amount of code released in RM. Policies was hardware support was (still) free and free code couldn't have dependency on charged for code

Basic design was from earlier 5-way SMP work but with nothing in microcode, but some of the software was restructured to be similar

	SJR/VM Release 5
	• Block Pre-paging (track previous pages)
	• SYSPAG
	• Single chain for eligible list
	• Simple group scheduling
	• Restructure IOS for performance & availability
	• CMS chained terminal I/O
	• Restructure CMS sysgen
	• Multiple file directories in shared segments

Much of the IOS reworked had been done for use by disk engineering & product test labs (bullet proof and never fail). They had previously tested MVS and found it to have 15min MTBF.

	SJR/VM Release 5.12
	• Extensive timing measurements
	• Runnable & non-runnable measured
	• Restructured Q3 controls
	• IOBLOK queued and service times
	• PAGE I/O queued and service times

Lots of things were being time-stamped and then various service & queued times accumulated for resource policy decisions. Also time-stamping was used in the IOS rewrite to implement MIH.

○		○
○	VM370 Release 6 (SEPP)	○
○		○
○	• CMS EDF	○
○	• Additional ECPS support for CMS	○
○	• S&Y directories in shared segment	○
○	• Uniprocessor V=R support	○
○		○
○		○
○		○
○		○
○		○
○		○
○		○
○		○
○		○

CMS enhanced filesystem supported minidisks formatted for 1k, 2k, and 4k record sizes.

	CPU	I/O	elapsed
4k/EDF	3.72	1958	82
PAM/EDF	3.41	3836	56
PAM/CDF	3.02	3790	52

Original CDF was 800 byte disk records. PAM/CDF reworked CDF for 4k page records. PAM/EDF modified 4k/EDF operation. There were also infrastructure changes to do contiguous allocation, delayed (block) writes and increased number of read records. Since operations were already page aligned and page operations, it eliminated enormous amount of I/O simulation overhead. CP could also re-organize operation sequence and chain multiple requests for operation service (as workload increased, benefits multiplied)



	<h2>VM/SP-HPO Release 1</h2>	
	<ul style="list-style-type: none"><li>• CP does block 3270 for multiple simulated line</li></ul>	
	<ul style="list-style-type: none"><li>• SMP reworked for single, non-SMP guest</li></ul>	
	<ul style="list-style-type: none"><li>• Support for hardware “protect”</li></ul>	
	<ul style="list-style-type: none"><li>• Significant increase in CP overhead masked by other changes</li></ul>	

Major customer opportunity was TPF on 3081. Originally 3081 was to only come in multiprocessor version and TPF didn't have multiprocessor support. VM/SMP support was reworked to increase the concurrent processing in a dedicated TPF environment (increased overhead but allowed more of CP overhead to execute concurrently with TPF). 3270 fullscreen write improvement somewhat masked this change ... however most CMS intensive SMP customers found overall throughput declined and TTY/non-3270 intensive customers found significant performance decline (my SJR/VM 5 CMS line writes accomplished pretty much the same, but for all customers).

## CP67-3.2 v. HPO2.5

machine	360/67	3081	ratio
MIPS			40-50
names	105	7000	66
users	80	320	4
channels	6	24	4
drums	12m	72m	*6
Page i/o	150	600	4



SYSPAG came from SJR/VM enhancing how the disk page & spool area definitions are handled (previously they had been strictly based on device)

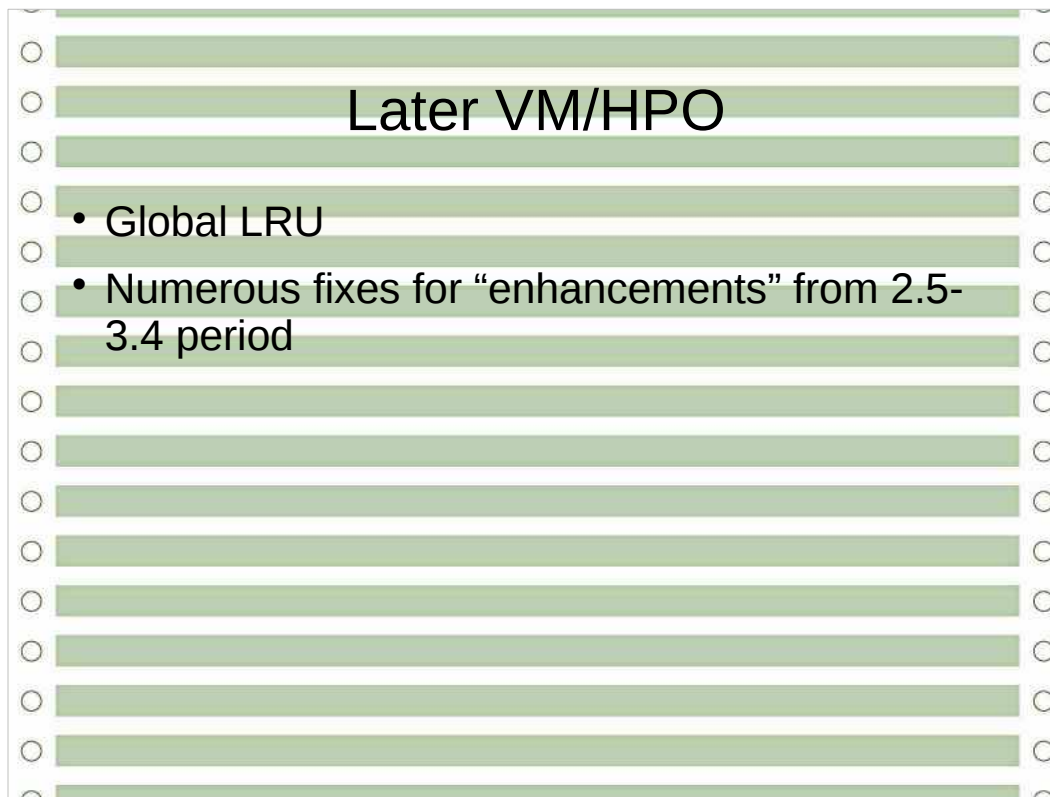
300 mills delay was degrading performance.

There was a specific problem that I had done in CSC/VM & SJR/VM that was in CP67 and I called VMDVBSY (which defined "long-wait" based on real device type). VM370 change resulted in "long-wait" being based on virtual device type. This was fine as long as they were similar. However, there was mismatch between virtual 3215 and real 3270. For every 3270 "enter" there would be three queue add/drops (when there should only be one)

HPO changes involved lots of twiddling that lacked extensive testing.

One was a change in HPO2.5 that would fail to reload floating point registers under certain circumstances (VM20536, HPO 2.5, 3.0, 3.2, & 3.4).

Other HPO problem allowed user runaway (like VM370 Release 1)



HPO2.5 corrupted global page replacement and taken out in HPO3.4. They had been claiming 80% of the code going into 3.4 was SYSPAG by Lynn Wheeler ... then I was told there were six OIAs for 3.4 code. I was then working with group putting global LRU back in (and correcting several other enhancements from the 2.5-3.4 period) showing better performance.

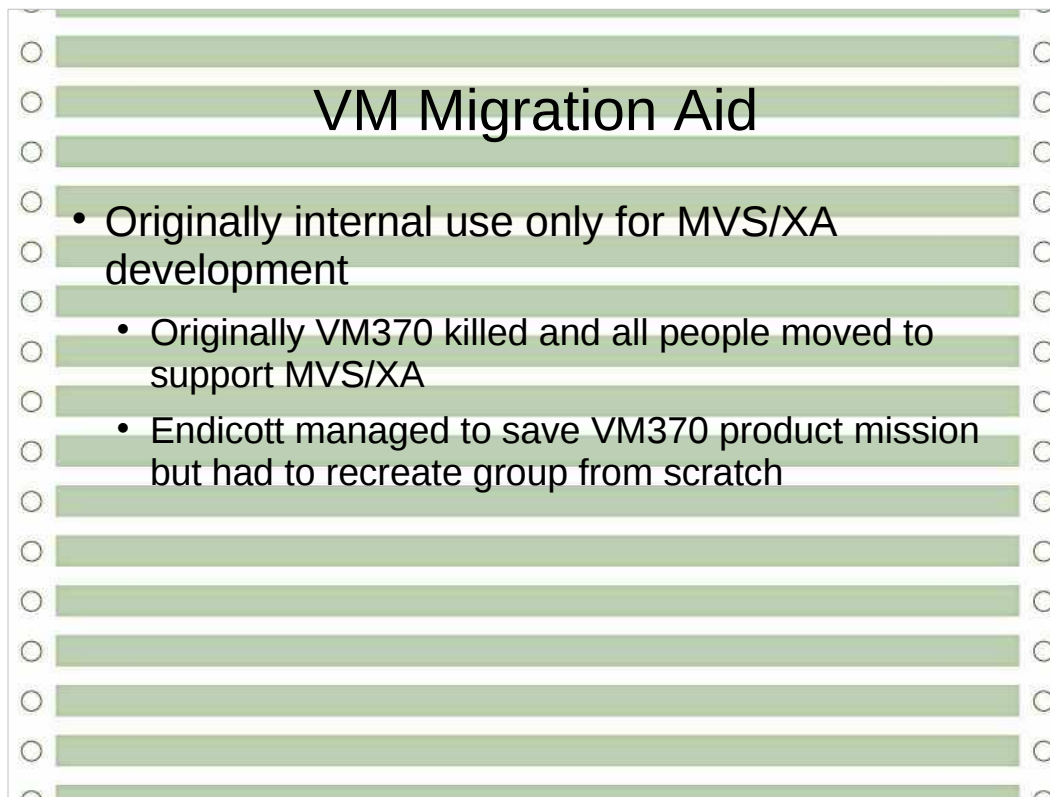
<http://www.garlic.com/~lynn/2011c.html#email860111>

<http://www.garlic.com/~lynn/2011c.html#email860119>

<http://www.garlic.com/~lynn/2011c.html#email860501>

and later reference

<http://www.garlic.com/~lynn/2011e.html#email870320>



Part of FS demise and mad rush to get items back into product pipeline, POK managed to convince corporate to kill vm370, shutdown development group and move all the people to POK to support MVS/XA development. Part of that was (internal only) 370/xa virtual machine test & development tool. 3081 SIE instruction was created for virtual machine tested, but it had severe performance issues since it had to be "paged" on the 3081

Eventually there was decision to release the internal tool as a "migration aid" (from MVS to MVS/XA) ... old email about enormous resources to cleanup the code for release. It had about 60% the performance of VM/HPO on the same platform.

Internally, one person in Rochester modified VM/HPO3.2 to support 370/xa and SIE instruction. The migration aid organization management contacted Rochester management to shut the person down and obliterate all evidence

<http://www.garlic.com/~lynn/2011c.html#email860121>

<http://www.garlic.com/~lynn/2011c.html#email861014>

The Migration aid was improved to have 80% the performance of VM/HPO

## Cluster z/VM

- Minor mention from Hillgang 2009 presentation
- From the Annals of Release No Software Before Its Time
- Internal US HONE datacenters (world-wide sales & marketing support) consolidated in silicon valley in mid-70s.
- Implemented load-balancing and fall-over across large loosely-coupled cluster with large shared disk farm.