

Introduction to Power 64-bit

Tom Cook

IBM

Topics

- Power was designed for the future
- Three key things to remember about 64-bit and the Power 970FX
- What is (and what isn't) 64-bit computing
- Mixing and matching - 64-bit processors and 32-bit software
- Running more 32-bit Applications using a 64-bit OS
- System Snapshots
- Leave 32-bit Applications 32-bit – example from Linux
- Embedded Solutions and 64-bit

Power was designed for the Future

- The original Power documents included 64-bit provisions
- Founders designed Power to last well into the future
- Ease of transition for applications to 64-bit proves Power has the edge in 64-bit computing

Thinking of Switching to 64-bit –
Worried it might hurt?



Solution: Don't do it all at once

While 64-bit is important -

Protecting 32-bit Software Investment is job #1

Important facts to take away from today's discussion:

1 64-bit Power processors today support two dynamic modes of addressing:

- 32-bit Mode
- 64-bit Mode

2 Proven binary compatibility with today's 32bit Applications

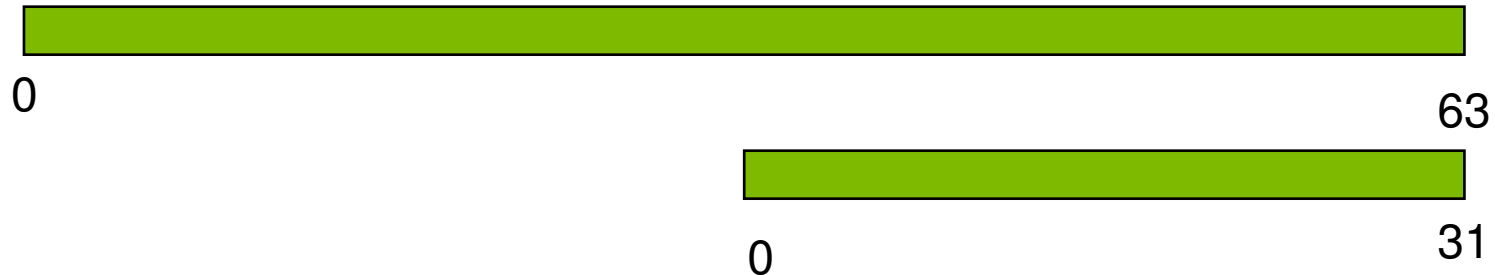
- 32bit Applications, System services, and Altivec/VMX enhanced code – No problem
 - No recompilations required
 - No penalty for running in 32-bit mode

3 Running a 64-bit Power processor with a 64-bit OS can help 32-bit applications with real memory constraints

64-bit Power processors are great 32bit Power CPUs

64-bit – When is a CPU 64-bit?

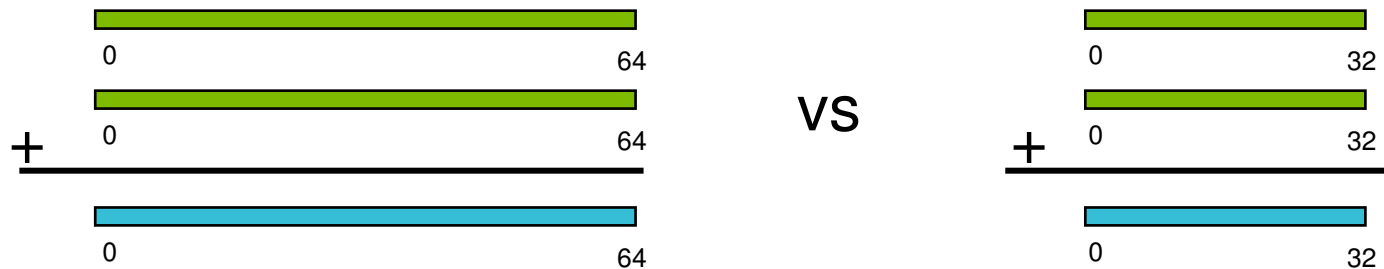
- **Hardware general purpose registers are 64-bits long**
- **64-bit integer and 64-bit logical operations executed with a single instruction**
- **Virtual addressing requires 64-bit pointers**



When do system designers turn to a 64-bit solution:

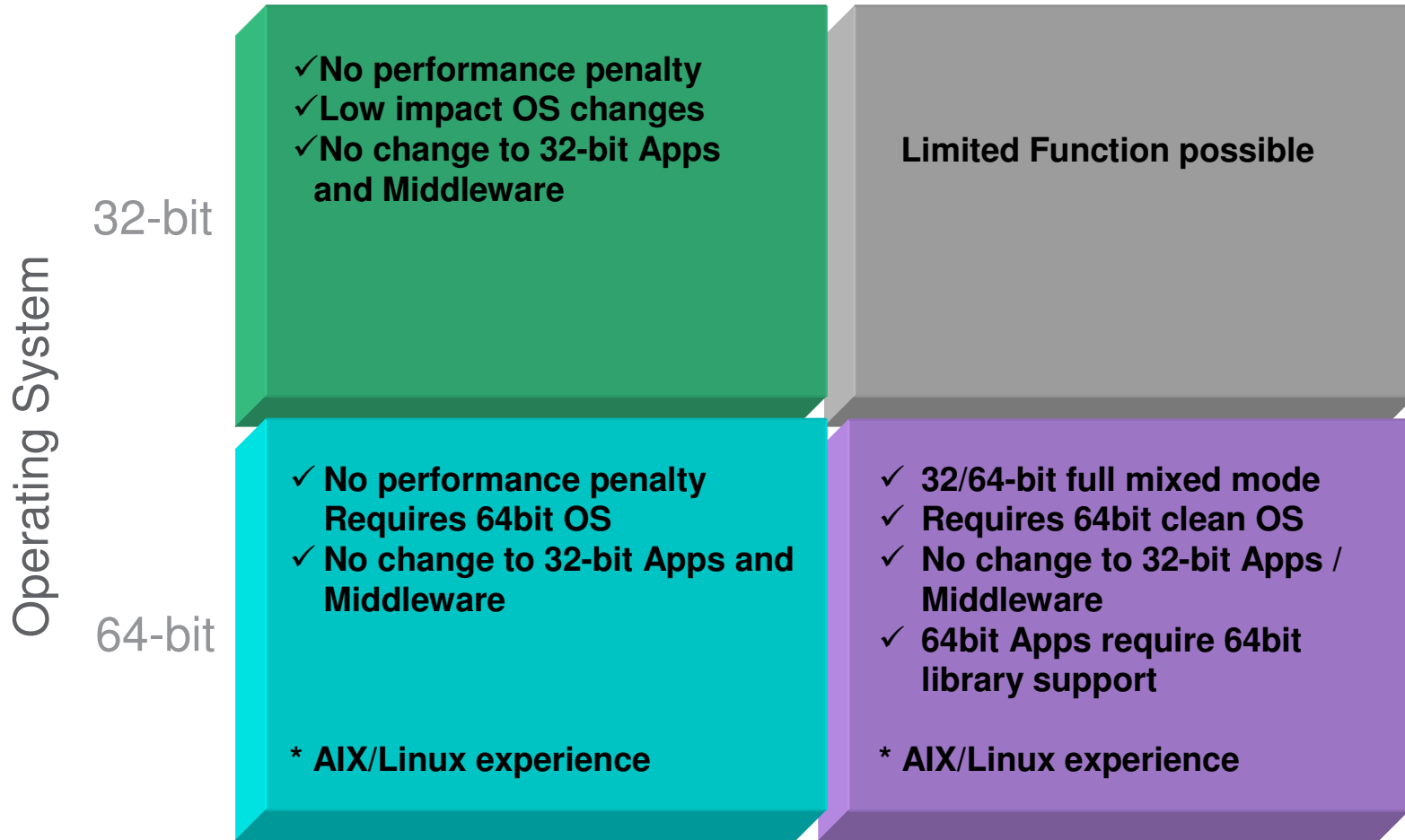
- Applications that need more virtual memory
 - 32-bit: Virtual limit is 4 Gigabytes for 32bit = 2^{32}
 - 64-bit: Virtual limit is 16 Exabytes for 64bit = 2^{64}
 - Note: Realistic virtual heap/CPU limits today are set lower than 2^{64}
- Applications that need more real/physical memory
 - Example: 970FX supports a real address limit of 2^{42} bits = **4 Terabytes**
 - 32-bit AND 64-bit applications take advantage of > 32-bit real/physical
- Applications that need Big Integer Math

Exabyte is 1,000 petabytes
Petabyte is 1,000 terabytes
Terabyte is 1,000 gigabytes



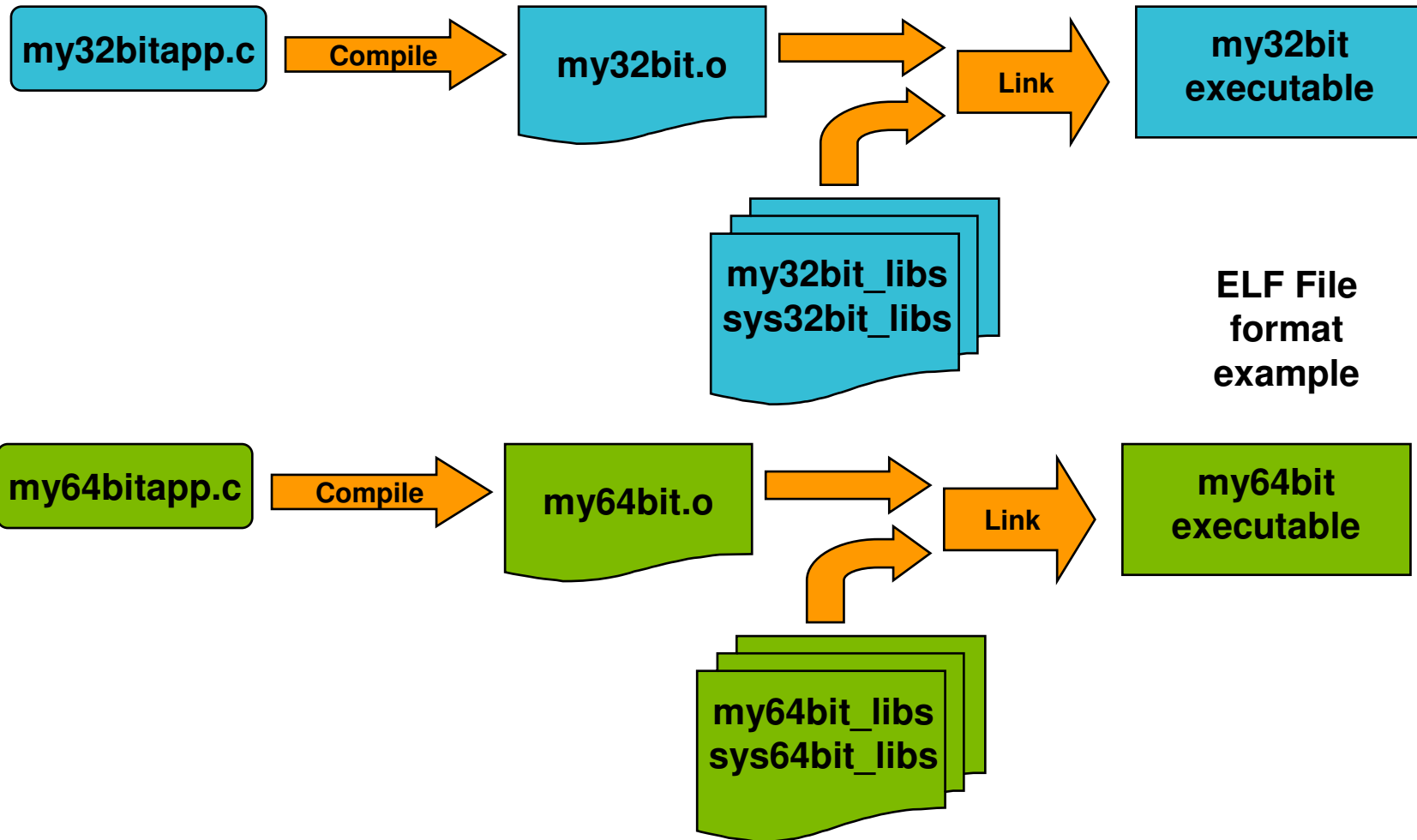
Sample 32-bit & 64-bit Operating Modes

32-bit Applications 64-bit

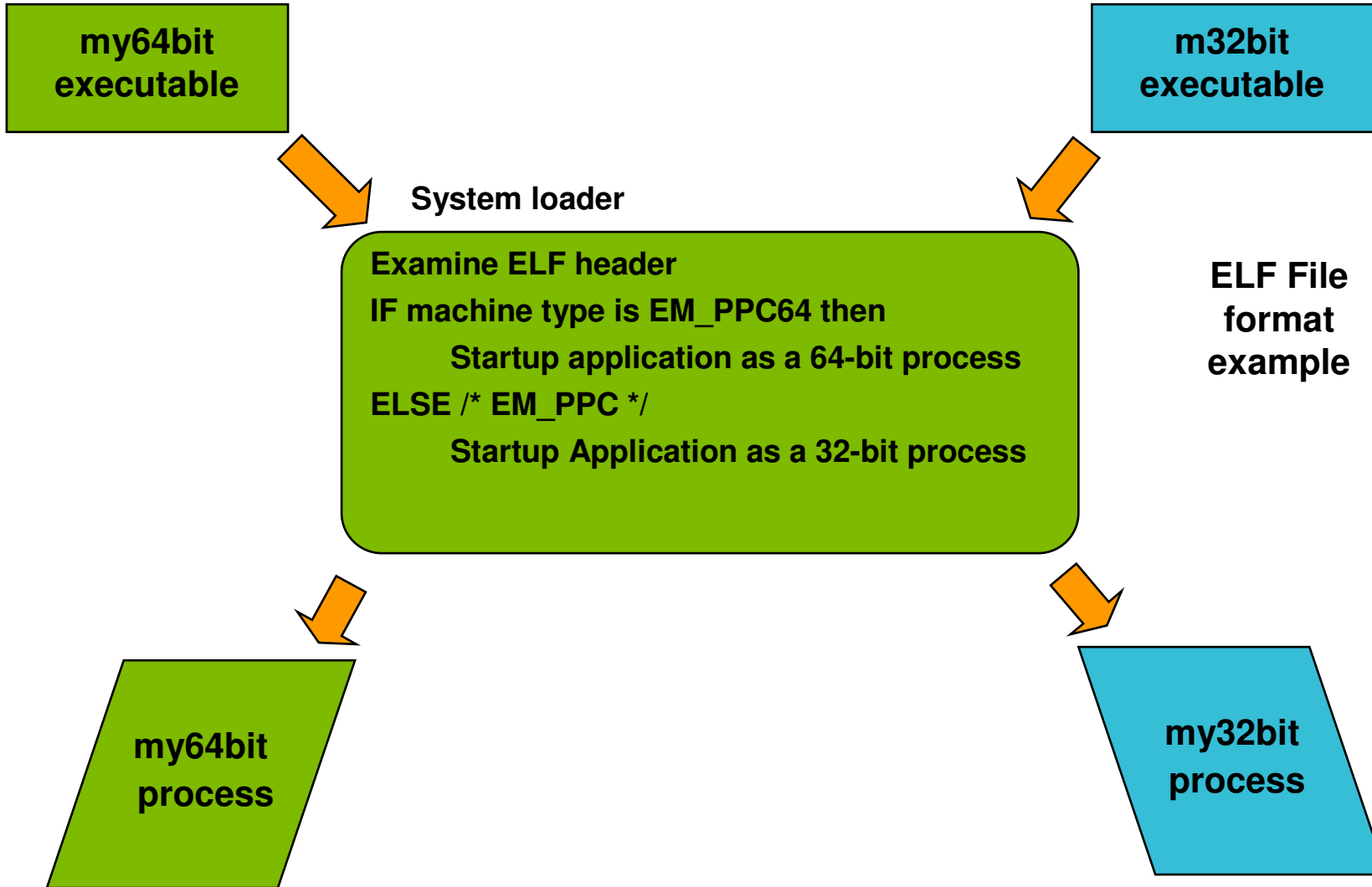


Scalable migration - move your applications to 64bit at your pace

Building 64-bit Applications vs 32-bit Applications

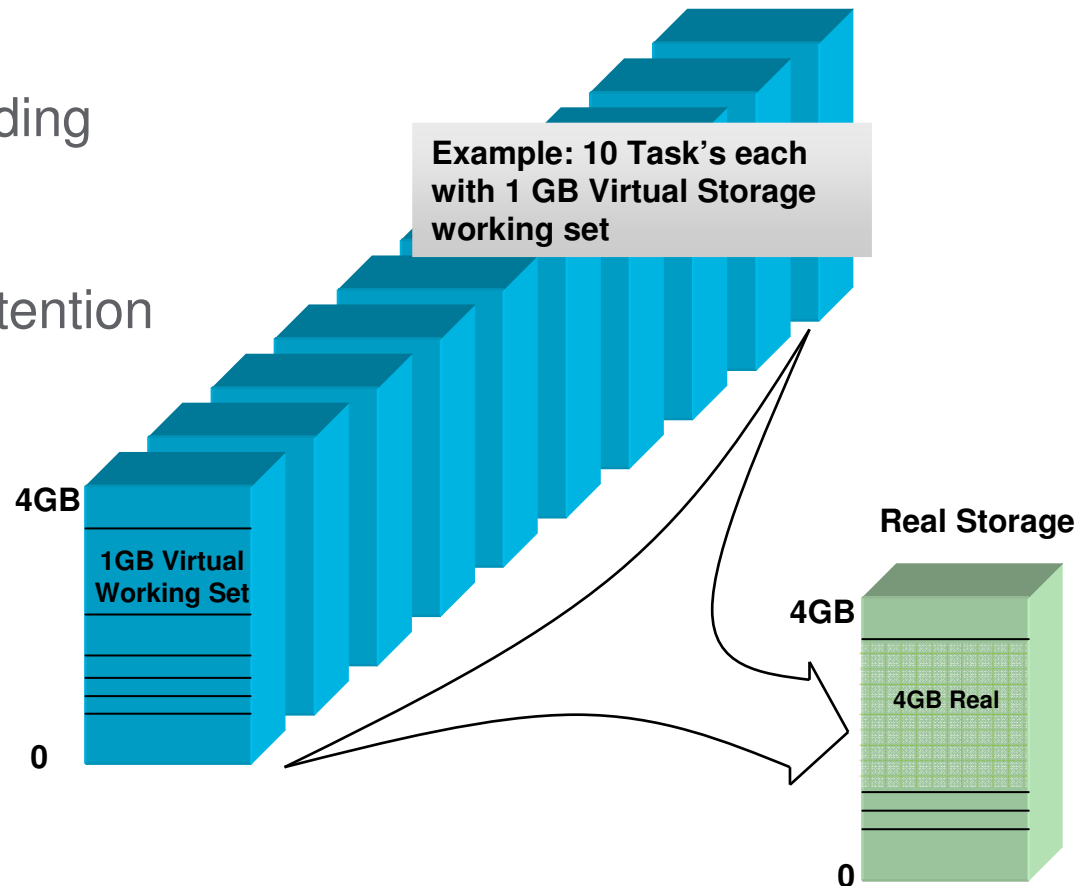


Running 64-bit Applications vs 32-bit Applications



32-bit CPU, 32-bit OS, 32-bit Applications

- ✓ More processes/threading
- ✓ Expanding heap sizes
- ✓ More real storage contention

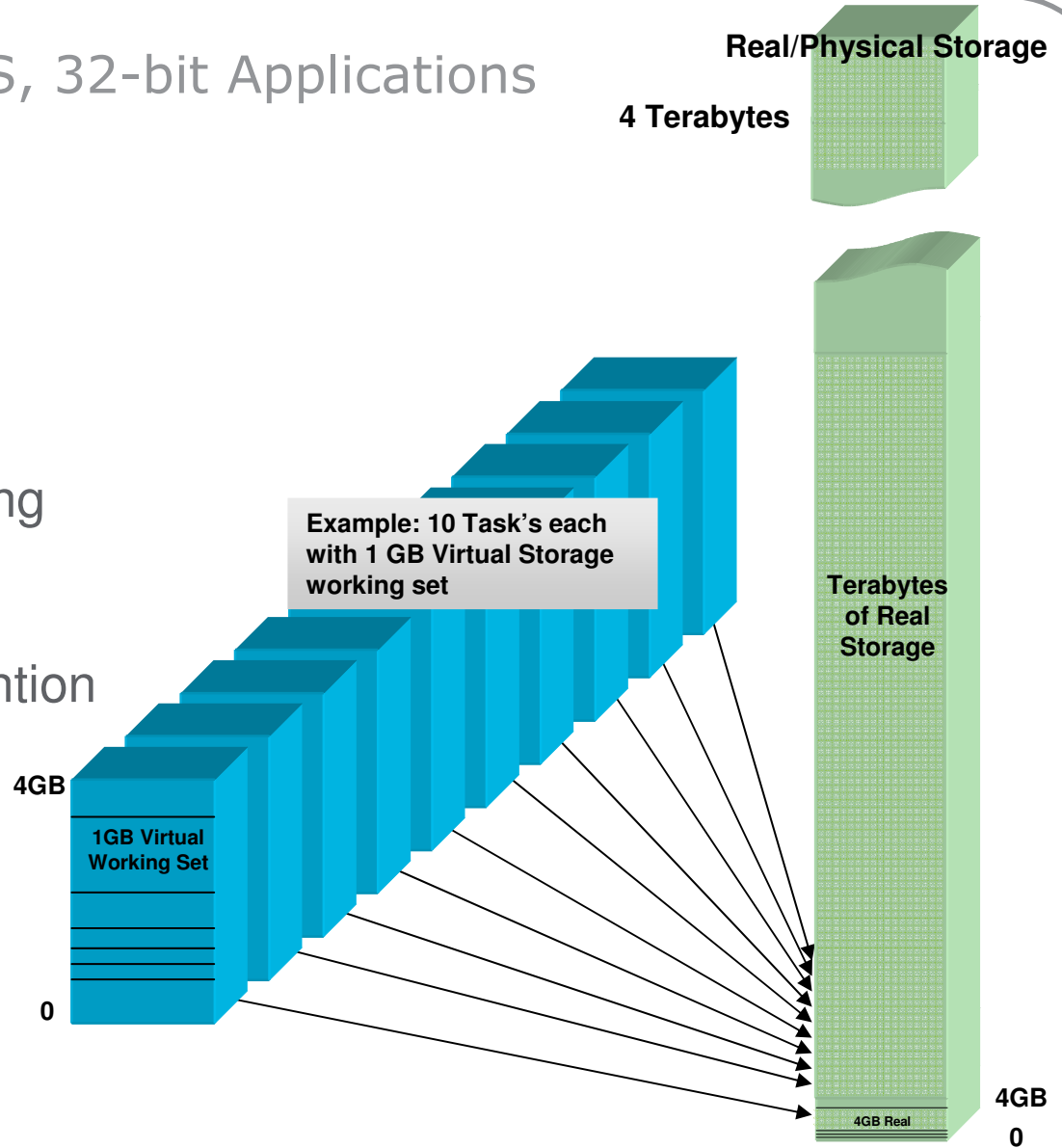


With tasks that have large working sets as pictured, each task contends for the 4GB of real **

64-bit CPU, 64-bit OS, 32-bit Applications

- Applications unchanged
- ✓ More processes/threading
- ✓ Expanding heap sizes
- ✓ Less real storage contention

64bit OS on Power can give each 32-bit task dedicated real storage

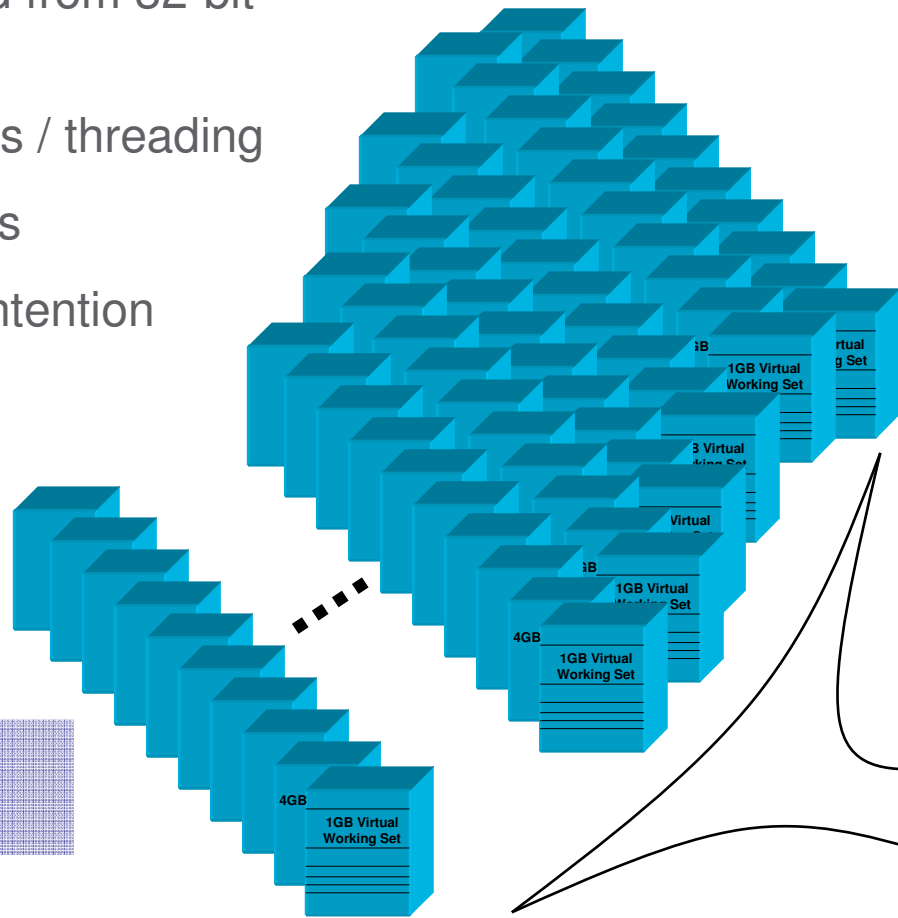


64-bit CPU, 64-bit OS, 32-bit Applications

Applications unchanged from 32-bit builds

- ✓ Many more processes / threading
- ✓ Expanding heap sizes
- ✓ Less real storage contention

Example: Many Task's each with 1 GB Virtual Storage working set



64bit OS with Power can give each 32-bit task dedicated real storage 4GB

Example of integer 32-bit Math vs 64 Math

long long x, y, z
z = x + y

64-bit Math*

ld r3,-32(r1)

ld r0,-24(r1)

add r3,r3,r0

std r3,80(r30)

64-bit Math / 32-bit ABI

ld r3,-32(r1)

ld r0,-24(r1)

add r3,r3,r0

mr r4,r3

srdi r3,r3,32

stw r3,80(r30)

stw r4,84(r30)

32-bit Math

lwz r3,64(r30)

lwz r4,68(r30)

lwz r5,72(r30)

lwz r6,76(r30)

addc r4,r4,r6

adde r3,r3,r5

stw r3,80(r30)

stw r4,84(r30)

Example of 64-bit integer usage for timer

64-bit Timer Operation

function_prolog(__kern_tb_read)

mfspir r5,tblr

std r5,0x0000(r3)

blr

function_epilog(__kern_tb_read)

(Works in 32-bit mode on
a 64-bit Power CPU)

32-bit Timer Operation

function_prolog(__kern_tb_read)

mfspir r5,tbur

mfspir r4,tblr

mfspir r8,tbur

cmp cr0,r5,r8

bne __kern_tb_read

stw r5,0x0000(r3)

stw r4,0x0004(r3)

blr

function_epilog(__kern_tb_read)

Example: WindRiver – Snapshot of 32-bit and 64-bit support

✓ **WindRiver's VxWorks - 32-bit only OS running on 970FX**

- Industrial strength high performance 32-bit RTOS

✓ **WindRiver compiler supports 64-bit operations**

✓ **WindRiver Workbench support 970FX in 32-bit mode AND 64-bit mode**

Green Hills Software - Snapshot of 970 32-bit and 64-bit support

- ✓ **Green Hills Integrity OS - 32-bit only OS running on 970FX**
 - Industrial strength high performance 32-bit RTOS
- ✓ **Green Hills Compiler Generates 64-bit Integer math**
 - ✓ GHS Integrity OS running in 32-bit mode supports 64-bit math

AIX – Snapshot of Power 64-bit support

- ✓ **Full support for 64-bit and 32-bit Applications**
 - ✓ **32-bit Applications NOT executing in emulation or partial emulation**
- ✓ **Kernel runs in 64-bit mode but changes processor mode to 32-bit to run 32-bit Applications**
- ✓ **Unified 32-bit and 64-bit libraries and location**

Linux – Snapshot of Power 64-bit support

- ✓ **Full support for 64-bit and 32-bit Applications**
 - ✓ **32-bit Applications NOT executing in emulation or partial emulation**
- ✓ **Kernel runs in 64-bit mode but changes processor mode to 32-bit to run 32-bit Applications**
 - How: Set 32-bit mode in SSR1 and execute rfid
- ✓ **Generally - separate 32-bit and 64-bit dynamic libraries**

```
tomcook@rs170a:~/> ls /
```

```
bin boot dev etc home lib lib64 media mnt opt proc root sbin srv tftpboot tmp usr var
```



Can I really run 32-bit Applications on a 64-bit Power processor

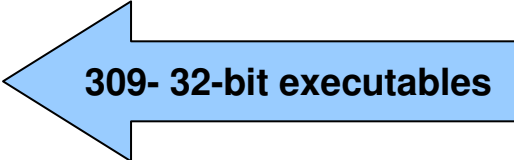
Yes! Even the system binaries on a 64-bit Linux system are mostly 32bit

/usr/sbin/zebra: ELF 32-bit MSB executable, Power, version 1 (SYSV), dynamically linked (uses shared libs), not stripped

/usr/sbin/zic: ELF 32-bit MSB executable, Power, version 1 (SYSV), dynamically linked (uses shared libs), stripped

```
tomcook:/> file /usr/sbin/* | grep "32" | wc
```

```
309 5727 46695
```



309- 32-bit executables

```
tomcook:/> file /usr/sbin/* | grep "64" | wc
```

```
1 17 142
```



Only 1 64-bit executable

Do I have to move most of my system applications to 64bit?

Less than 4% of the user commands needed to move to 64-bit

/usr/bin/znew:	Bourne shell script text
/usr/bin/zoo:	ELF 32-bit MSB executable, Power, version 1 (SYSV), dynamically linked (uses shared libs), stripped
/usr/bin/zsh:	symbolic link to ../../bin/zsh
/usr/bin/zsoelim:	ELF 32-bit MSB executable, Power, version 1 (SYSV), dynamically linked (uses shared libs), stripped

```
tomcook: /> file /usr/bin/* | grep "32" | wc
```

```
787 14498 126444
```

```
tomcook: /> file /usr/bin/* | grep "64" | wc
```

```
28 465 4215
```



787- 32-bit executables



Only 28 - 64-bit executables

Another Linux 64-bit implementation with Less 64-bit code

64-bit Kernel and very few 32-bit systems Applications

```
[tomcook@yhpc /]$ ls
```

```
bin dev home lib lost+found mnt proc sbin sys tmp var  
boot etc initrd lib64 misc opt root selinux tftpboot usr
```

Dual libs – 32-bit and 64-bit

```
[tomcook@yhpc /]$ cat /proc/version
```

```
Linux version 2.6.10-1.ydl.1g5-smp (root@build.terraplex.com) (gcc version 3.3.3 (Yellow  
Dog Linux 3.3.3-16.ydl.7)) #1 SMP Mon Feb 7 10:18:19 MST 2005
```

```
[tomcook@yhpc /]$ file /boot/vmlinux-2.6.10-1.ydl.1g5-smp
```

```
/boot/vmlinux-2.6.10-1.ydl.1g5-smp: ELF 64-bit MSB executable, cisco 7500, version 1  
(SYSV), statically linked, stripped
```

64-bit Kernel

```
[tomcook@yhpc /]$ file /usr/sbin/* | grep "64" | wc
```

```
4 76 704
```

```
[tomcook@yhpc /]$ file /usr/bin/* | grep "64" | wc
```

```
21 411 3777
```

```
[tomcook@yhpc /]$
```

Performance and Production code 32-bit and 64-bit

Benchmarks for 970 Family run in 32-bit mode

✓ **SPECint and SPECfloat built in 32-bit mode**

- 32-bit mode with 64-bit Integer math

✓ **EEMBC benchmarks for 970 built with Green Hills 32-bit compiler – others with GCC in 64-bit mode**

✓ **Dhrystone benchmark run in 32-bit mode using IBM Compilers**

Basic Rule – Don't move your Application to 64-bit unless you must

Subtle inefficiencies of 64-bit code

- 64-bit code is generally larger in size
 - Pointers are 64 bits long vs 32 bits long
 - Larger code means more i-cache misses
- Stack sizes are larger
 - When running in 64-bit mode, all 64 bits of registers are saved on the stack

Applications that benefit from a 64-bit CPU

- ✓ Imaging – Very large files can fit into virtual memory
 - Animation, film editing/rendering
 - Physical environment simulation
- ✓ Databases – Very large tables memory mapped
- ✓ Life sciences – Large data size, 64-bit math
- ✓ Storage products – Virtualization of file systems
- ✓ Networking – Large routing tables
- ✓ Soft switching – More compute bandwidth

Power 64-bit CPUs support today's 32-bit applications while providing expansion to 64-bit

Mapping Big Files using 64-bit Virtual Addressing

32-bit Mode requires chopping up large files



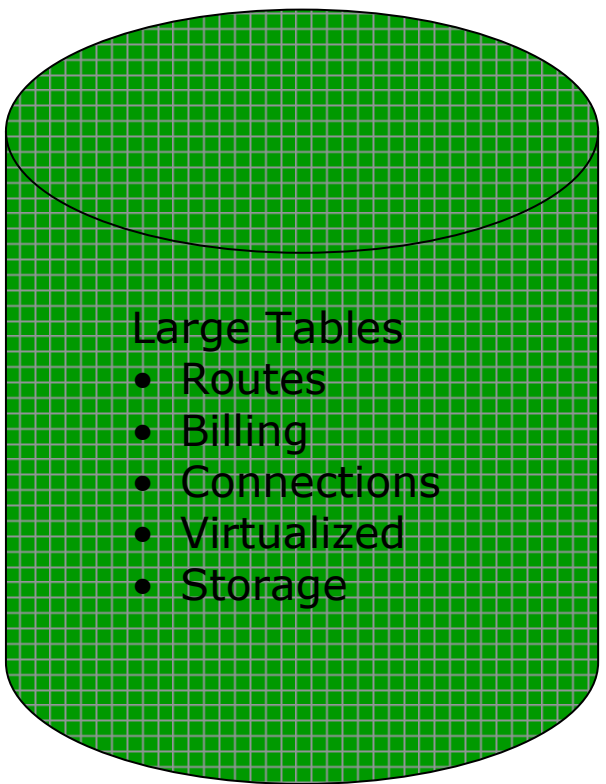
64-bit Mode lets you map large files in virtual

- No need to slice and dice your files so they fit

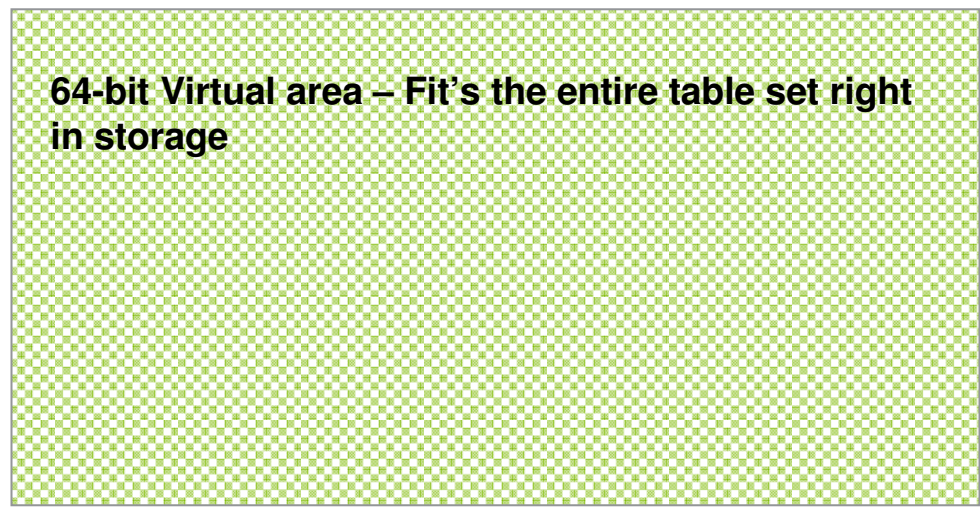


Even if you do not have a large amount of Physical memory, you can still map large data chunks into virtual memory

Mapping Big Tables using 64-bit Virtual Addressing



 32-bit Virtual area – need to manually load table segments



Large files aren't just for the enterprise any more.....

Conclusion

- Power was designed for the future
- Move your applications at your own pace – Power is friendly to mixed mode computing
- Scale your systems, not your applications as step one
- Identify key applications that need Big Math or Big Spaces – Move them first
- Enterprise, Desktop, and Embedded opportunities for 64-bit are here now

End