2015中国DPDK开发者大会
China DPDK Summit 2015

Presented By:

中国移动 China Mobile | 中国电信 CHINA TELECOM | HUAWEI | intel | ZTE

DPDK on POWER: A New Enhanced Packet Processing Architecture

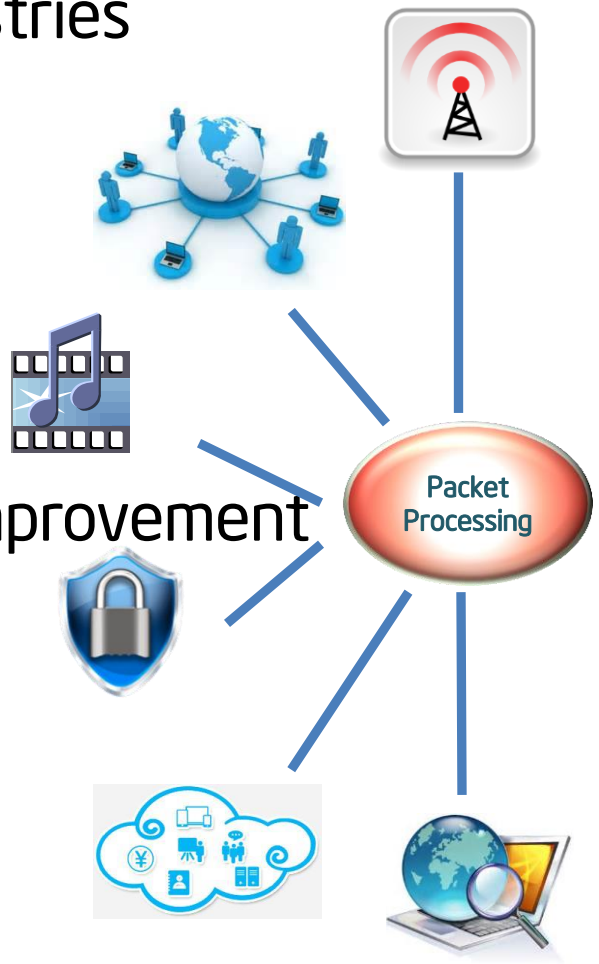
Dr. Zhu Chao
IBM Research – China
2015.04.21

Agenda

1. DPDK on multi-architecture
2. IBM POWER architecture
3. Migration experience
4. Summary and future work

Why do we want a multi-architecture DPDK?

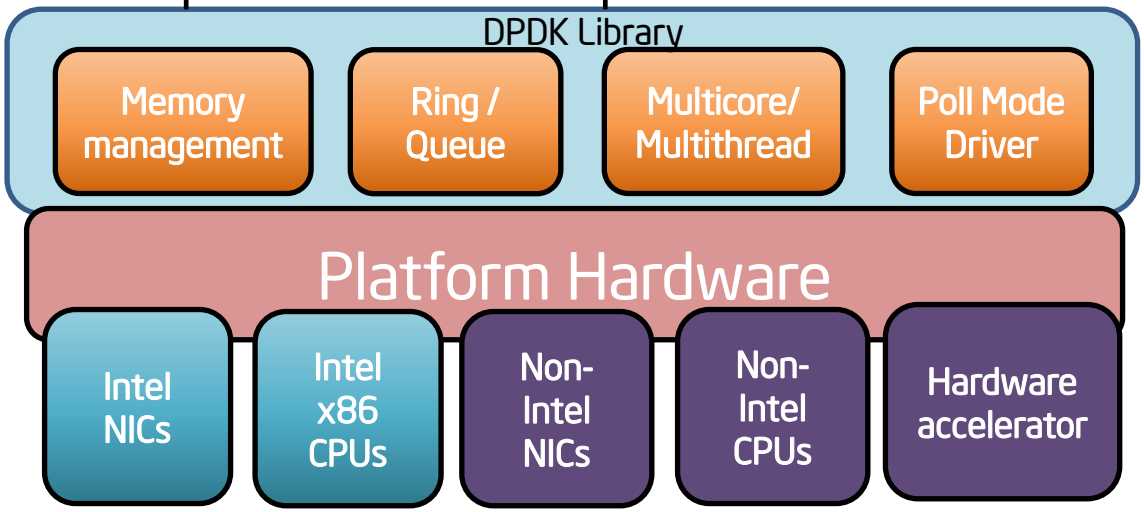
- Fast packet processing is crucial for industries
  - Telecom operator / manufacturer - NFV
  - Cloud / data center operator
  - Security companies
  - Video sharing companies
  - Government
- DPDK leads to significant performance improvement
  - Pure software-based optimization
  - General purpose computing architecture
  - Intel Xeon E5 L3 forwarding[1]: 10Mpps/core
  - 6X faster than Linux native stack



1. http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/communications-packet-processing-brief.pdf

Why do we want a multi-architecture DPDK?

- Trend: DPDK should be hardware-independent
  - Common optimization techniques for difference CPU architecture
  - Common optimization techniques for difference NICs



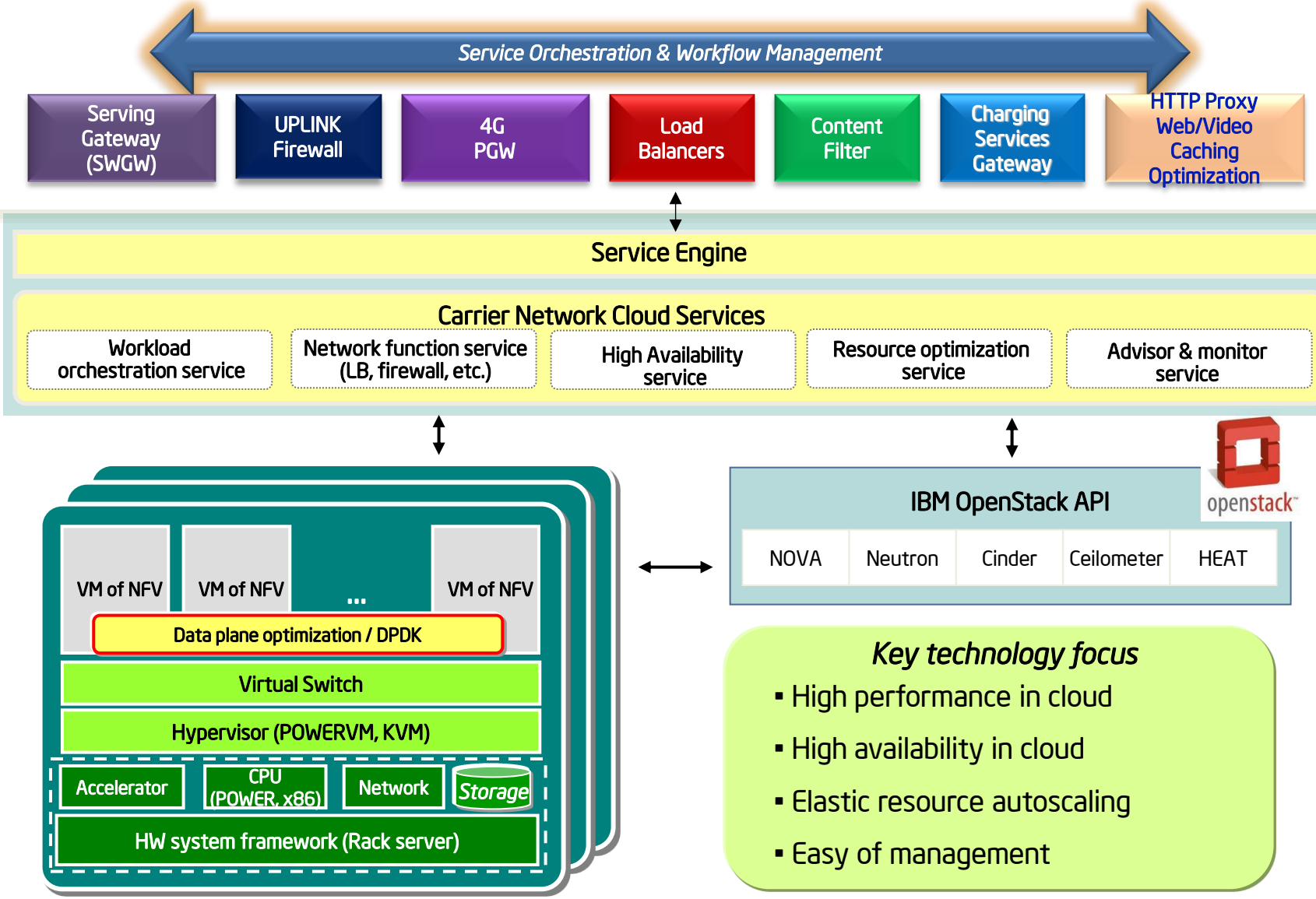
intel
ARM
Power
PowerPC
Imagination TECHNOLOGIES
MIPS TECHNOLOGIES
Mellanox TECHNOLOGIES
EMULEX
BROADCOM
Connecting everything

NFV on IBM POWER – IBM cloud platform to support NFV

- NFV is a big trend and opportunity
  - IBM invested to it from 2010
  - The TD-LTE NFV prototype was shown on IMIC2013 and MWC 2014
  - POWER processor is already involved in the NFV prototype

NFV on IBM POWER – IBM cloud platform to support NFV

Data Plane Optimization plays a vital role in NFV



Key technology focus

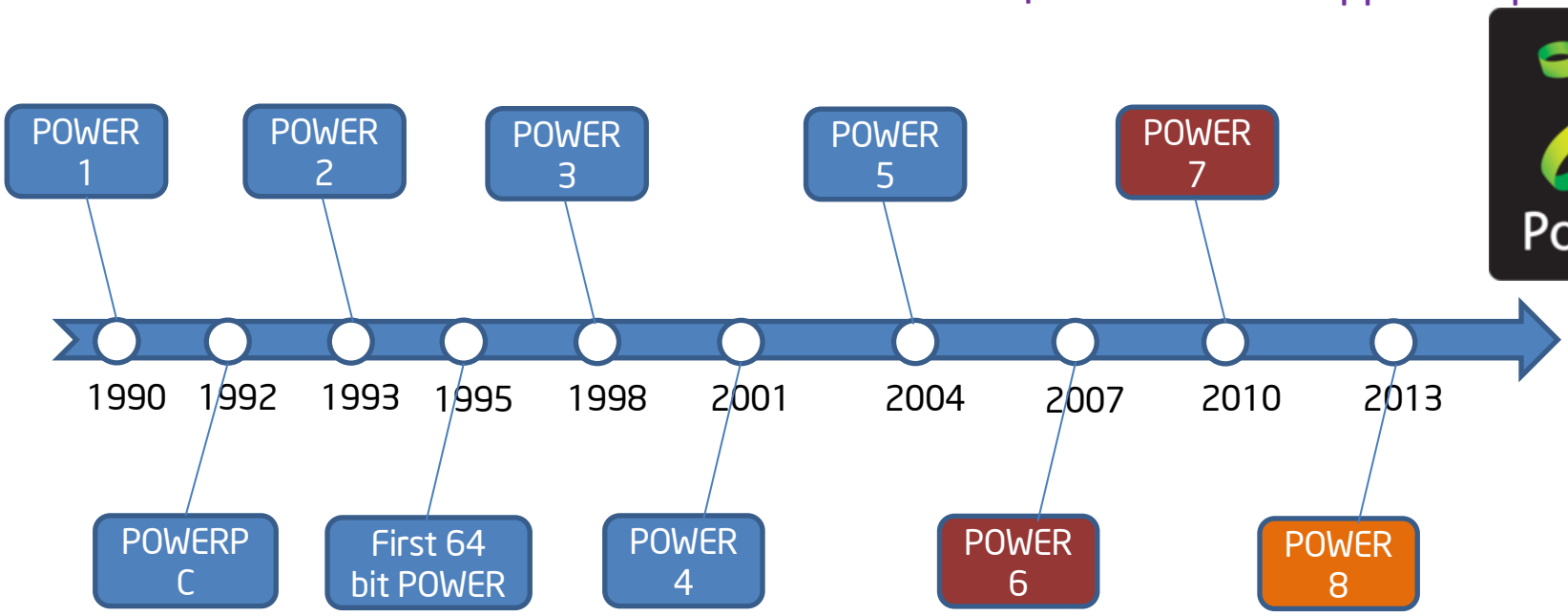
- High performance in cloud
- High availability in cloud
- Elastic resource autoscaling
- Easy of management

POWER Architecture

POWER: Performance Optimization with Enhanced RISC

PowerPC

- IBM Watson
- Mars rovers
- Embedded processor
- Apple computer



Power™

OpenPOWER™

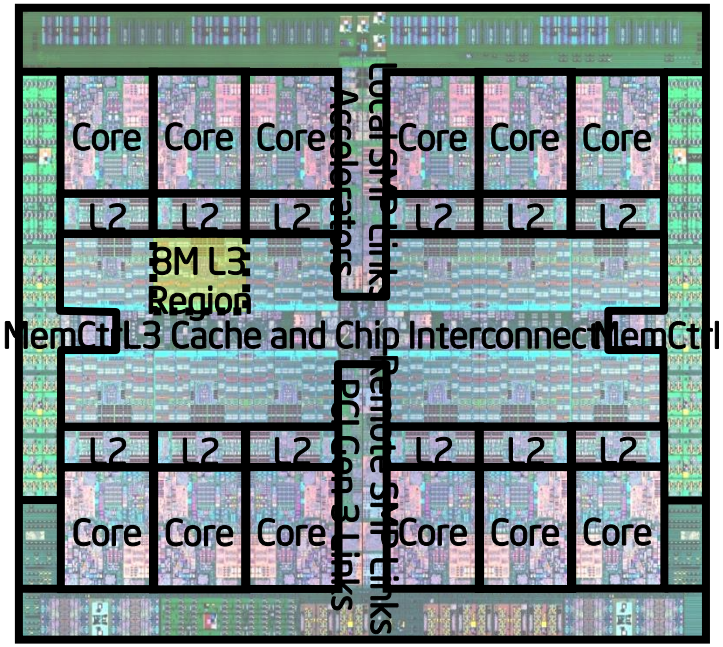
- Modern POWER: From POWER6
- OpenPOWER: From POWER8, provide customized processor and platform
- Open source: DPDK

IBM POWER8 v.s. Intel E5 v3

Enhanced platform for high performance packet processing

CPU	Intel E5 v3	POWER8
Max Freq.	3.7GHZ	5GHZ
Max Pcore	18	12
Max Lcore	36	96
Cache	L2: 256K LLC: 2.5MB/Core	L2: 512 K L3: 8MB/Core eDRAM L4: Max 128 MB off chip
Max Memory size	768GB	1TB
Max Memory speed	68 GB/s BW	230 GB/s sus BW
On-chip PCIe Gen3	40 lanes	48 lanes
Chip interconnection	QPI: 9.6 GT/s x 2=19.2GT/s	150 GB/s x 12 seg = 3.6TB/s
Accelerator IF.	N/A	CAPI: Coherent Accelerator Processor Interface

- High performance packet processing
  - Computation power
  - Fast I/O and memory bandwidth
- P8 has excellent computation ability



Migration experiences – Challenges

- DPDK is tightly coupled with x86 architecture
  - Compiler micros doesn't help
- Low level code migration
  - X86 and POWER ISA doesn't have 1-1 mapping
  - X86 hardware specific instructions
    - rdtsc, cpuid, sse, etc.
  - POWER ISA changes from POWER7
    - Half-word and byte atomic operations
- Performance optimization
  - Optimization should leverage CPU features
- Virtualization technology
  - Hypervisor

Migration experiences – Considerations

- Start from April. 2014.
- Minimize the migration efforts
  - Migrate from a common Linux version
  - Both x86 and POWER platform support Linux
  - The compiler can be GCC
- Physical host first, virtual guest later
- Porting first, optimizing later
- Migrate POWER 64 bit first

Migration experiences – Know Architecture Programming Difference

- The basic difference: ISA
  - Most of the difference can be hided by compiler
- The compiler difference
- The byte order difference
- The data width difference
- Data alignment differences
- Hardware differences

Migration experiences – Find Out Architecture-Specific Code

- ISA related code
  - ASM code
    - asm volatile {...}
- Compiler related code
  - Architecture specific compiling options
    - DPDK/config
    - DPDK/mk
    - Gcc: -march is not supported on POWER
  - Compiler built-in micros

Migration experiences – Find Out Architecture-Specific Code

- Platform specific features

Architecture	X86_64	PPC_64
Endianness	Little endian	Big endian / little endian
Cache line size	64 Byte	128 Byte
Huge page size	4KB/2MB/1GB	64KB/16MB/16GB
Logic cores / processor	Max 36 Lcores	Max 96 Lcores
Vector instructions	SSE	VMX / VSX
CPU flag register	Yes	N/A
Time register	TSC register	Time base register
HW CRC	Yes	N/A

Migration experiences – Find Out Architecture-Specific Code

Architecture specific files
/lib/librte_eal/common/include/rte_atomic.h
/lib/librte_eal/common/include/rte_prefetch.h
/lib/librte_eal/common/include/rte_byteorder.h
/lib/librte_eal/common/include/rte_cycles.h
/lib/librte_eal/common/include/rte_cpuflags.h
/lib/librte_eal/common/include/rte_spinlock.h
lib/librte_eal/common/eal_common_cpuflags.c
/lib/librte_eal/common/include/rte_memcpy.h
Building system files
mk/rte.cpuflags.mk
mk/machine
mk/arch/
config/
lib/librte_eal/common/Makefile
Plaform specific files
lib/librte_eal/linuxapp/eal/eal.c
lib/librte_eal/linuxapp/eal/eal_memory.c
lib/librte_eal/common/eal_common_memzone.c
lib/librte_eal/common/include/rte_memory.h
lib/librte_eal/common/include/rte_memzone.h
lib/librte_eal/linuxapp/eal/include/exec-env/rte_kni_common.h
lib/librte_lpm/rte_lpm.h
lib/librte_lpm/rte_lpm.c
lib/librte_lpm/rte_lpm6.c
app/test/test_malloc.c
app/test/test_memzone.c
app/test/test_cpuflags.c
app/test-pmd/config.c

Migration experiences – Add POWER related operations

- ASM code related operations
  - Atomic operations
    - Inc, dec, test and set, compare and set, barrier, etc.
  - Prefetch operations
    - POWER doesn't have prefetch inst. for different cache level
  - Spinlock operations
  - POWER itself has different ISA
    - Compatibility
- OS Kernel has a good reference of the ASM implementation
- GPL license / BSD license
- POWER ISA book provides a good guide
    - www.power.org

Migration experiences – Add POWER related operations

- Example: 32bit atomic inc

```
/* x86 code*/
#define MPLOCKED        "lock ; "

rte_atomic32_inc(rte_atomic32_t *v)
{
asm volatile(
MPLOCKED
"incl %[cnt]"
: [cnt] "=m" (v->cnt)   /* output */
: "m" (v->cnt)          /* input */);
}
```

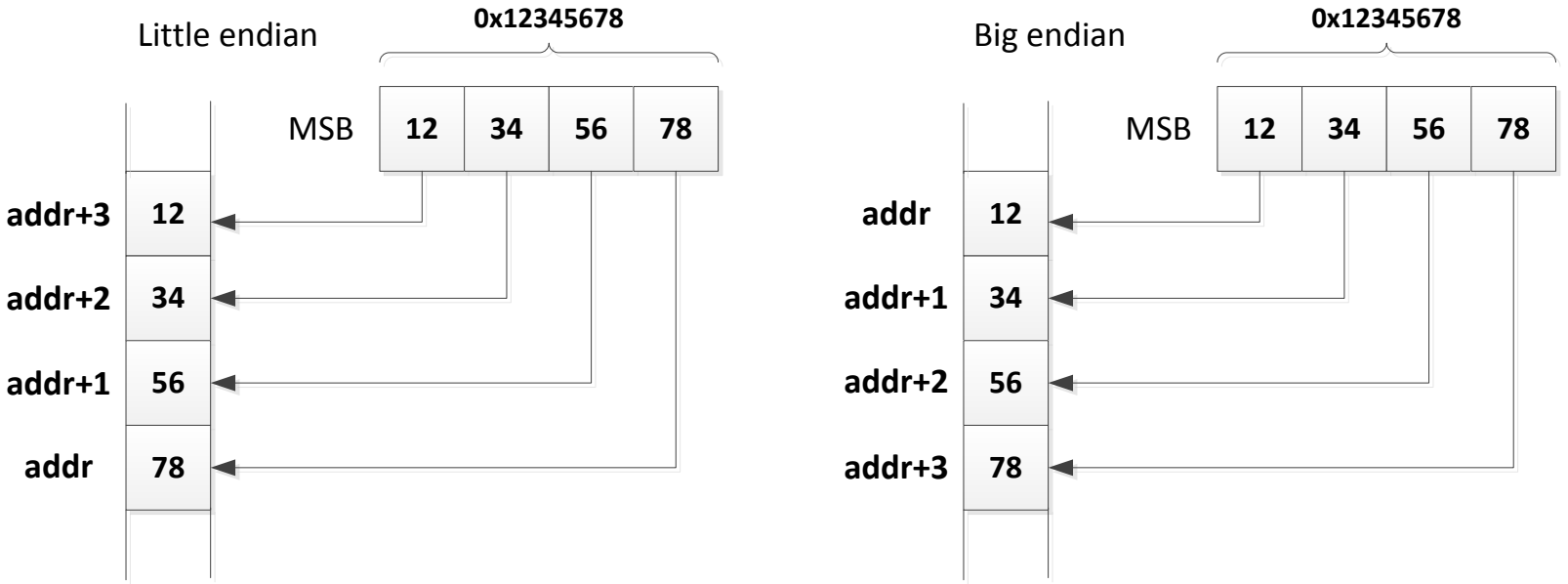
```
/* PPC code*/
rte_atomic32_inc(rte_atomic32_t *v)
{
int t;
asm volatile(
"1: lwarx     %[t],0,%[cnt]\n"
"    addic     %[t],%[t],1\n"
"    stwcx.    %[t],0,%[cnt]\n"
"    bne- 1b\n"
: [t] "=&r" (t), "=m" (v->cnt)
: [cnt] "r" (&v->cnt), "m" (v->cnt)
: "cc", "xer", "memory");
}
```

Migration experiences – Add POWER related operations

- Building systems for POWER
  - Add POWER specific configuration file
    - Config/defconfig_ppc_64-power8-linuxapp-gcc
    - Define CONFIG_RTE_ARCH to “ppc_64”
    - Turn off the compilation of un-migrated libraries
  - Add POWER specific compilation flags
    - mk/arch/ppc_64 and mk/machine/power8
    - Remove unsupported CFLAGS

Migration experiences – Add POWER related operations

- Little Endian and big endian
  - Different CPU endian: x86: Little endian POWER: Big / little endian
  - Determined by the memory address of most significant byte
  - ntoh(), hton() for network packet processing



Migration experiences – Add POWER related operations

- Little Endian and big endian
  - byte swap functions are used to implement the APIs
    - rte_cpu_to_le/be_16/32/64()
    - rte_le/be_to_cpu_16/32/64()
  - x86 has specific instructions to do byte swapping
    - xchgb / bswap
  - POWER needs to use shift and bit operators

```
/* x86 little endian */
rte_arch_bswap32(uint32_t _x)
{
   register uint32_t x = _x;
   asm volatile ("bswap %[x]"
                 : [x] "+r" (x)
    );
  return x;
}
```

```
/* POWER big endian */
rte_arch_bswap32(uint32_t _x)
{
return ((_x >> 24)
   | ((_x >> 8) & 0xff00)
   | ((_x << 8) & 0xff0000)
   | ((_x << 24) & 0xff000000));
}
```

Migration experiences – Add POWER related operations

- Little Endian and big endian
  - structure define

```
/* x86 little endian structure*/

struct rte_lpm_tbl8_entry {
uint8_t next_hop;
uint8_t valid       :1;
uint8_t valid_group :1;
uint8_t depth       :6;
};
```

```
/* POWER big endian structure*/

struct rte_lpm_tbl8_entry {
uint8_t depth       :6;
uint8_t valid_group :1;
uint8_t valid       :1;
uint8_t next_hop;
};
```

- Use RTE_LITTLE_ENDIAN / RTE_BIG_ENDIAN micro at compile time
- Get endian information from Linux / code
  - #include <endian.h> endian related micros

Migration experiences – Add POWER related operations

- Data type and alignment
  - Most of the data type and alignment are the same
    - x86_64 v.s. PPC 64
    - x86_32 v.s. PPC 32
  - data type *char* has different default type
    - x86: signed char POWER: unsigned char
  - data type *long double* has different size
    - x86 32bit: 96 bit 64bit: 128 bit
    - POWER 32 bit: 128 bit 64bit: 128 bit (64 bit if use -qnoldb128 in XLC compiler)
  - data alignment is required to avoid possible data corruption
    - Long double - x86 32 bit : 4B align 64bit: 16B align
    - Long double - POWER 32 bit : 8B align 64bit: 8B align

Migration experiences – Add POWER related operations

- CPU Flags
  - Determine CPU features at run time
    - Intel processors have CPUID instruction to get CPU supported features
    - There is no such kind of instructions in POWER architecture
  - /proc/cpuinfo is not feasible
    - portability
    - Doesn't have detail information on POWER
  - auxiliary vector from kernel
    - ELF loader parse the vector
    - HWCAP/ HWCAP2 from /proc/self/auxv

```
/proc/cpuinfo of IBM POWER8
processor  : 0
cpu        : POWER8E (raw), altivec supported
clock      : 3325.000000MHz
revision   : 2.1 (pvr 004b 0201)
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
stepping        : 4
cpu MHz         : 2099.805
cache size      : 15360 KB
physical id     : 0
siblings        : 12
core id         : 0
cpu cores       : 6
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb
rdtscp lm constant_tsc arch_perfmon pebs bts rep_good xtopology
nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2
ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm ida arat xsaveopt pln pts
dts tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms
bogomips        : 4199.61
...
```

/proc/cpuinfo of Intel CPU

Migration experiences – Add POWER related operations

- Auxiliary vector for POWER
  - Architecture specific definition in asm/cputable.h

HWCAP		HWCAP2	
PPC_FEATURE_32	0x80000000	PPC_FEATURE2_ARCH_2_07	0x80000000
PPC_FEATURE_64	0x40000000	PPC_FEATURE2_HTM	0x40000000
PPC_FEATURE_601_INSTR	0x20000000	PPC_FEATURE2_DSCR	0x20000000
PPC_FEATURE_HAS_ALTIVEC	0x10000000	PPC_FEATURE2_EBB	0x10000000
PPC_FEATURE_HAS_FPU	0x08000000	PPC_FEATURE2_ISEL	0x08000000
PPC_FEATURE_HAS_MMU	0x04000000	PPC_FEATURE2_TAR	0x04000000
PPC_FEATURE_HAS_4xxMAC	0x02000000	PPC_FEATURE2_VEC_CRYPTO	0x02000000
PPC_FEATURE_UNIFIED_CACHE	0x01000000		
PPC_FEATURE_HAS_SPE	0x00800000		
PPC_FEATURE_HAS_EFP_SINGLE	0x00400000		
PPC_FEATURE_HAS_EFP_DOUBLE	0x00200000		
PPC_FEATURE_NO_TB	0x00100000		
PPC_FEATURE_POWER4	0x00080000		
PPC_FEATURE_POWER5	0x00040000		
PPC_FEATURE_POWER5_PLUS	0x00020000		
PPC_FEATURE_CELL	0x00010000		
PPC_FEATURE_BOOKE	0x00008000		
PPC_FEATURE_SMT	0x00004000		
PPC_FEATURE_ICACHE_SNOOP	0x00002000		
PPC_FEATURE_ARCH_2_05	0x00001000		
PPC_FEATURE_PA6T	0x00000800		
PPC_FEATURE_HAS_DFP	0x00000400		
PPC_FEATURE_POWER6_EXT	0x00000200		
PPC_FEATURE_ARCH_2_06	0x00000100		
PPC_FEATURE_HAS_VSX	0x00000080		
PPC_FEATURE_PSERIES_PERFMON_COMPAT	0x00000040		
PPC_FEATURE_TRUE_LE	0x00000002		
PPC_FEATURE_PPC_LE	0x00000001		

Migration experiences – Add POWER related operations

- Cache line size

```
#ifndef RTE_CACHE_LINE_SIZE
#define RTE_CACHE_LINE_SIZE 64
#endif
```

```
In mk/arch/ppc_64/rte.vars.mk:
CPU_CFLAGS  ?= -DCACHE_LINE_SIZE=128
```

- Huge page size

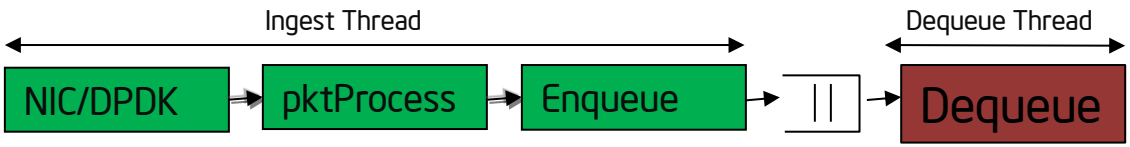
```
#define RTE_MEMZONE_2MB
0x00000001
#define RTE_MEMZONE_1GB
0x00000002
enum rte_page_sizes {
   RTE_PGSIZE_4K = 1 << 12,
   RTE_PGSIZE_2M = RTE_PGSIZE_4K << 9,
   RTE_PGSIZE_1G = RTE_PGSIZE_2M <<9
};
```

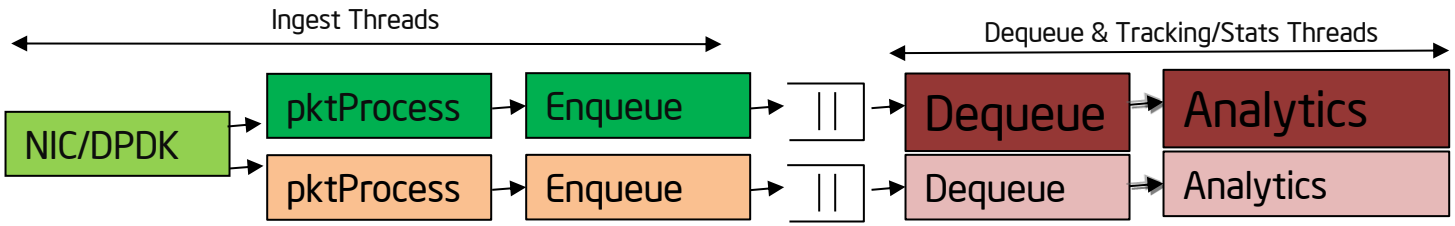
```
#define RTE_MEMZONE_1GB
0x00000002
#define RTE_MEMZONE_16MB
0x00000100
#define RTE_MEMZONE_16GB
0x00000200
enum rte_page_sizes {
            RTE_PGSIZE_4K = 1ULL << 12,
            RTE_PGSIZE_2M = 1ULL << 21,
            RTE_PGSIZE_1G = 1ULL << 30,
            RTE_PGSIZE_64K = 1ULL << 16,
            RTE_PGSIZE_16M = 1ULL << 24,
```

The huge page sizes are not overlapped

DPDK for POWER Use Case – Big data

- IBM InfoSphere Streams 'network sniffing' application
  - IBM POWER8 + Mellanox CX3-Pro NICs
    - POWER8 (3.3GHZ, 12 cores) 2 Socket, 2U NEBS Server
    - Mellanox CX3-Pro 2 ports NIC with 10GBE link
  - Summary of ingest thread work for each incoming packet:
    - Pull the packet from the NIC queue
    - Parse layer 2 & 3 headers
    - Read & hash all payload bytes.
    - Send a summary of key packet data downstream for further processing.





Current Migration Status

- Two sets of patches for PPC64 BE were submitted and were migrated to DPDK upstream v1.8
  - Accepted in Nov. 2014
  - Split architecture specific files
  - PPC64 specific files
- PPC64 LE migration is in progress

Current Migration Status

Libraries / Functions	Migration status
Malloc library	OK
Ring library	OK
Mempool library	OK
Mbuf library	OK
Timer library	OK
EAL library	OK
Poll mode driver	Support Mellanox CX3-pro PMD
HASH library	OK
LPM /ACL/QoS libraries	Not compiled
Vmware / Xen related	Not supported on POWER
KVM / QEMU related	Compiled but not tested on POWERKVM
POWERVM	OK

Future Work

- DPDK EAL migration for little endian POWER8
- Full EAL library migration
- Poll mode drivers for different NICs on POWER
- Virtualization support on POWER

Free OpenPower Cloud - SuperVessel

SuperVessel
WeChat group

www.ptopenlab.com

QQ group: SuperVessel

Thanks