# OpenPOWER: Reengineering a server ecosystem for large-scale data centers

Michael Gschwind
IBM Power Systems

# OpenPOWER is about choice in large-scale data centers


- build workload optimized solutions
- use best-of-breed components from an open ecosystem

- collaborative innovation in open ecosystem
- with open interfaces

- delivered system performance
- new capabilities instead of technology scaling

# Competing Forces Are Shaping the Data Center Landscape

*Need for Standardization*

*Need for Innovation*


**Hardware Development**

- ⇨ Consumer Scale Volumes
- ⇨ DIY servers

**Software Development**

- ⇨ Known Software Stack
- ⇨ Higher Programmer Productivity

**“X86 Everywhere”**

**Hardware Development**

- ⇨ Optimize for Servers
- ⇨ Increase Value & Differentiation

**Software Development**

- ⇨ Portability & Platform Choice
- ⇨ Workload Optimization

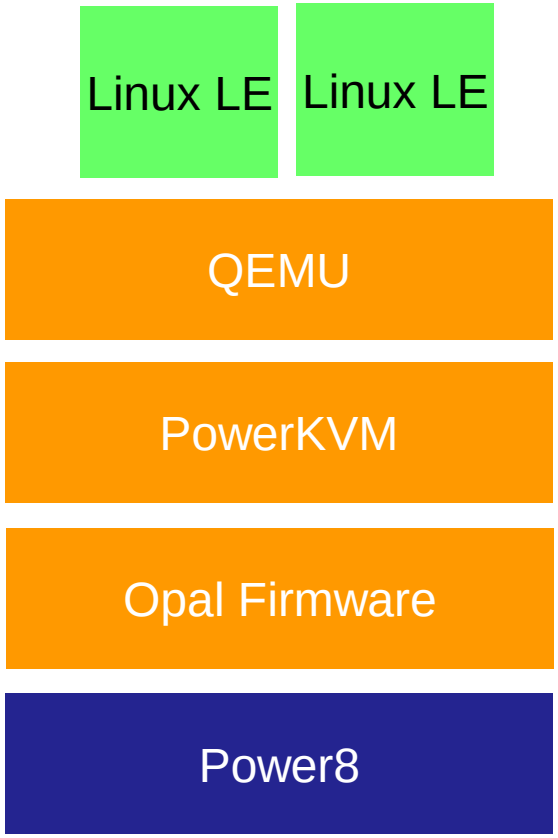
**One Size Does Not Fit All**

# OpenPOWER unchains scale out data centers

- A consortium of data center suppliers and operators
- Jointly create vibrant open ecosystem for data centers
- Expand options for large-scale data center computing
- Software stacks as true measure of ecosystem

# OpenPOWER ecosystem framework

- Enable rich ecosystem of hardware vendors
  - Standardized hardware interfaces
  - ⇨ Common, open firmware interfaces
- Open source system software stack
  - Data center operators rely on tuning SW stack
  - Enable server ODM vendors to create offerings
  - ⇨ Operating environment built on Linux and KVM
- Simplify porting of scale-out data center applications
  - Application source code dependences
  - In-storage data base formats
  - Exploit I/O and accelerators originally designed for PCs
  - ⇨ Little-endian data format and programming interfaces



# A new Open Power Linux environment

- OpenPOWER is not the traditional Power Linux with a new name
  - Significant discontinuity and fresh start
  - No Binary Compatibility to Legacy Power Linux
    ⇨ new environment “ppc64le”
  - New: Firmware, Hypervisor, data layout, source code, ELFv2 ABI

- What changes for application developers?
  - Byte order
  - New ABI
  - Vector programming API

- Extremely rapid turn-around from decision to develop to deployment
  - First discussions in March 2013, Linux distro builds starting October 2013
  - >40000 packages built within short time

# The New OpenPOWER Application Binary Interface (ABI)

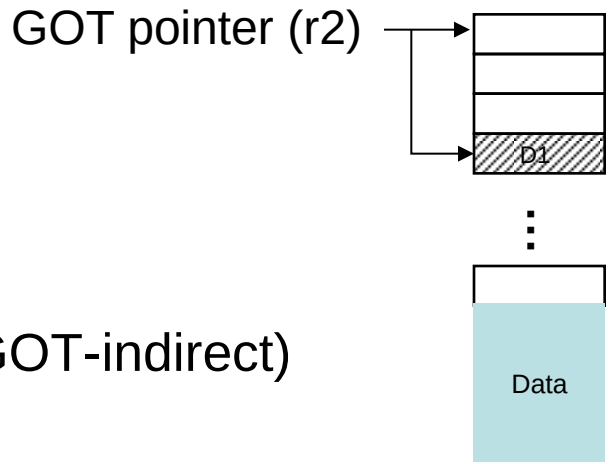
- Starting point: PPC64 / AIX ABI
  - Established, tested production code
  - Commonality and maintenance across LE, BE and AIX where feasible
  - Minimum disruption for tooling: GCC, XL, Java, LLVM, libffi, PyPy, …
- Define new capabilities as delta over baseline
  - Align with the broader ecosystem
  - Create hardware optimization opportunities / synergies
  - Optimize for modern code patterns
    - More classes, abstraction
    - Shorter function lengths
    - More indirect calls
- **If it ain't broken, don't fix it!**

# Optimizing Data Access with the Global Offset Table (GOT)

- Initialize pointer to GOT without functions descriptors
  - Dual entry points to re-use GOT on local call
- Optimize GOT pointer update on cross-module calls
  - Schedule GOT pointer save in caller
- Expand addressing range with “Medium Code Model”
  - Exploit Fusion and avoid GOT overflow code
- Optimize GOT in main module
  - Generate non-PIC code and resolve symbols at link time

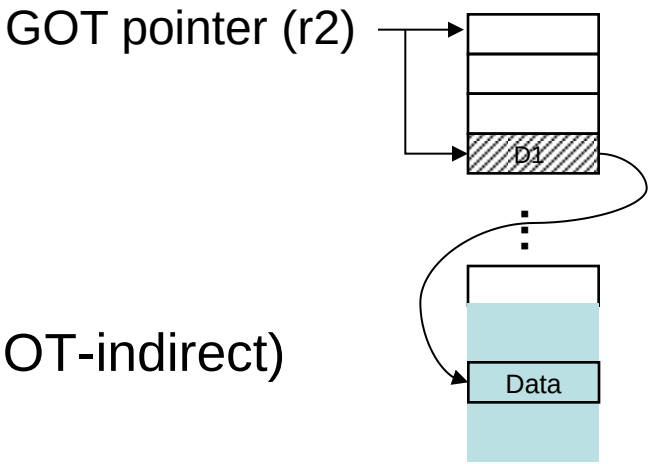
# Global Offset Table (GOT)

- Global Offset Table: data dictionary
  - Holds addresses for global data
- The GOT pointer points to data dictionary
  - load data address for indirect access (GOT-indirect)
  - load data directly (GOT-relative)
- Each module (shared library) has a different GOT
  - Cross-module calls save/restore old and load new GOT pointer



# Global Offset Table (GOT)

- Global Offset Table: data dictionary
  - Holds addresses for global data

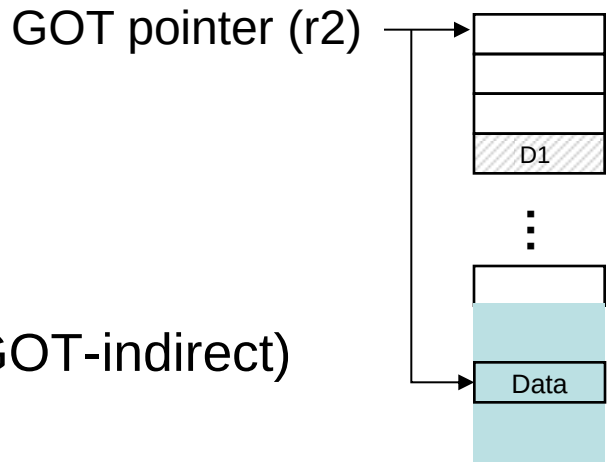



- The GOT pointer points to data dictionary
  - ➔ load data address for indirect access (GOT-indirect)
  - load data directly (GOT-relative)

- Each module (shared library) has a different GOT
  - Cross-module calls save/restore old and load new GOT pointer

# Global Offset Table (GOT)

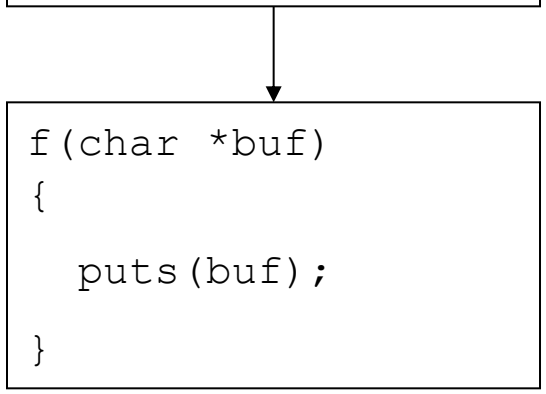
- Global Offset Table: data dictionary
  - Holds addresses for global data
- The GOT pointer points to data dictionary
  - load data address for indirect access (GOT-indirect)
  - ➔ load data directly (GOT-relative)
- Each module (shared library) has a different GOT
  - Cross-module calls save/restore old and load new GOT pointer



# Establishing the GOT pointer

- Two entry points for each function
  - "Local EP": shared GOT in same module
  - "Global EP": initialize new GOT for module
  - Symbol table points to Global EP
- Direct call provides current GOT pointer (in r2)
  - Linker resolves local call to Local EP
  - Linker resolves extern call to PLT stub
    - Extern call resolved by dynamic linker
    - PLT stub performs indirect call to function
- Indirect calls to Global EP
  - Function entry address in r12

```
f: ; r12 points to func entry
  addis r2, r12, (.TOC.-l0)@ha
  addi r2, r2, (.TOC.-l0)@l
.localentry f, 2
```

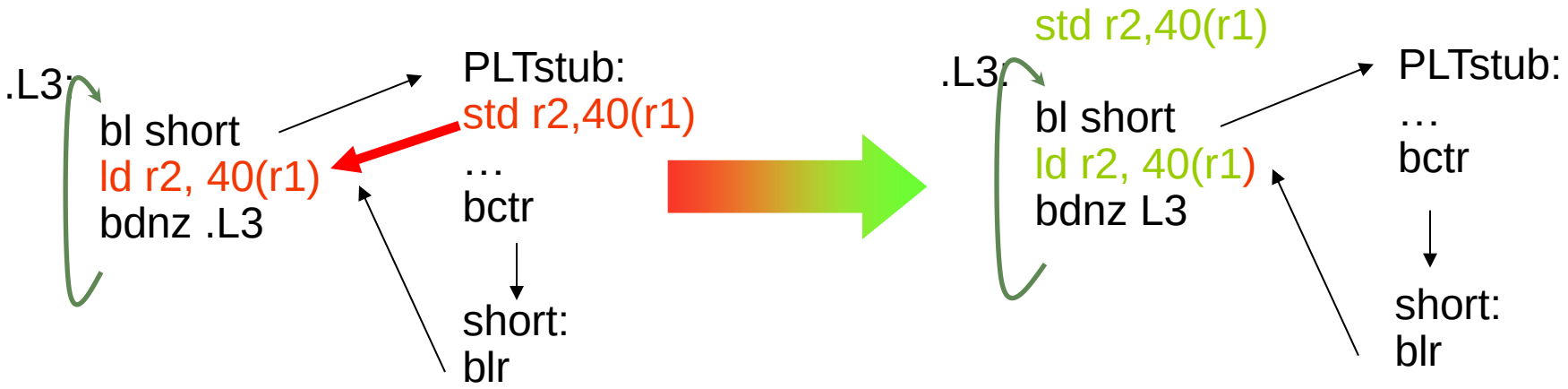


# Optimizing GOT pointer save and restore

- Compiler collaborates with linker
  - Reduce GOT management overhead
- Optimizing placement of GOT pointer save points for external calls
  - Compiler schedules a placeholder for a GOT pointer save
  - Static linker populates placeholder with GOT save instruction **if and only if** an external function call is made

# Call cost reduction with scheduled GOT store

- Remove repeated store of GOT pointer
- Avoid forwarding of in-flight store to dependent load



# Medium Code Model

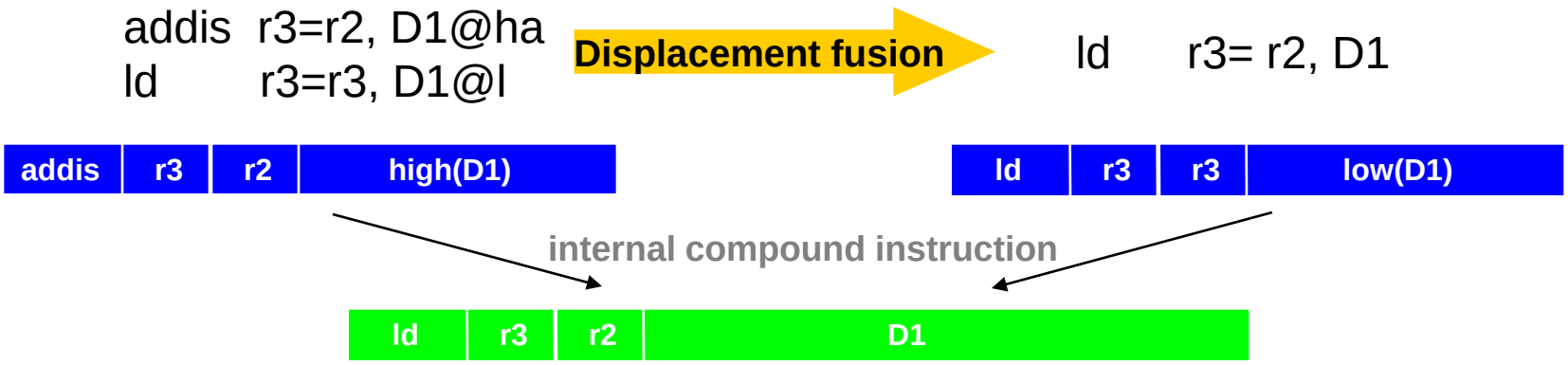
- "Medium code model" addresses growing data size
  - Expand GOT data dictionary to up to 4GB
  - Memory access with 32b displacement

- The size of the data dictionary is constrained by displacement range
  - Fixed width RISC ISA constrains the displacement to 16b
    - 64KB "natural" dictionary size (8k variables)
  - Dictionary overflow handling can penalize performance

- "Medium Code Model" as default: facilitate organic application growth
  - Enable applications to grow beyond 8K to 500M variables per module
  - "Beyond RISC" using Displacement Fusion capability in Power8

# Beyond RISC: Displacement Fusion

- "Fusion" allows processor to build **internal compound instruction**
  - Combine multiple instructions into internal compound instruction
  - CISC addressing range while retaining RISC fixed-width advantage
- Compound instruction executes as a single hardware operation
  - Fewer resources, shorter latency
  - Improved performance for programs with large data sets



# Function Call Improvements

- Parameter passing
  - Pass/return more parameters in registers
- Streamline stack frame
  - Allocate parameter save area only when required
- More descriptive object file info
  - More precise DWARF, Reloc's, and ELF format flags
  - Improve future ABI extensibility

# Register usage for function arguments

- Goal: Reduce abstraction penalty (“same performance as builtin types”)
  - OO languages wrap basic data types in a class
  - Previous Power ABI passes most classes via GPRs
  - … and returns most class results in memory
    - Other common ABIs pass and return class in memory
- Goal: Pass each data type in “natural” register
  - Integer parameters ⇨ general purpose registers
  - Floating point parameters ⇨ floating point registers
  - Vector parameters ⇨ vector registers

# Register usage for function arguments

- Pass more class types as register parameters
  - ⇨ Solution: homogeneous float/vector aggregates
  - – Up to 8 members passed in natural registers
- Return function results in same register(s) as first input parameter
  - – Aggregates returned in multiple registers
  - – Homogenous float and vector aggregates in float and vector registers
- Significant improvement for abstract data type handling
  - – 3x-4x performance improvement for invocations of class vertex<float>
  - – Class std:complex<> matches native _Complex type performance
  - – ~1%-5% performance improvement for many OO codes

# ABI performance

- Unchanged performance in programs with
  - Large functions / subroutines
  - Extensive inlining
- Faster performance in programs with:
  - Large GOT with many global variables
  - Calls to small functions
  - Pointer calls including virtual function calls.
  - Functions with abstract data types
  - …

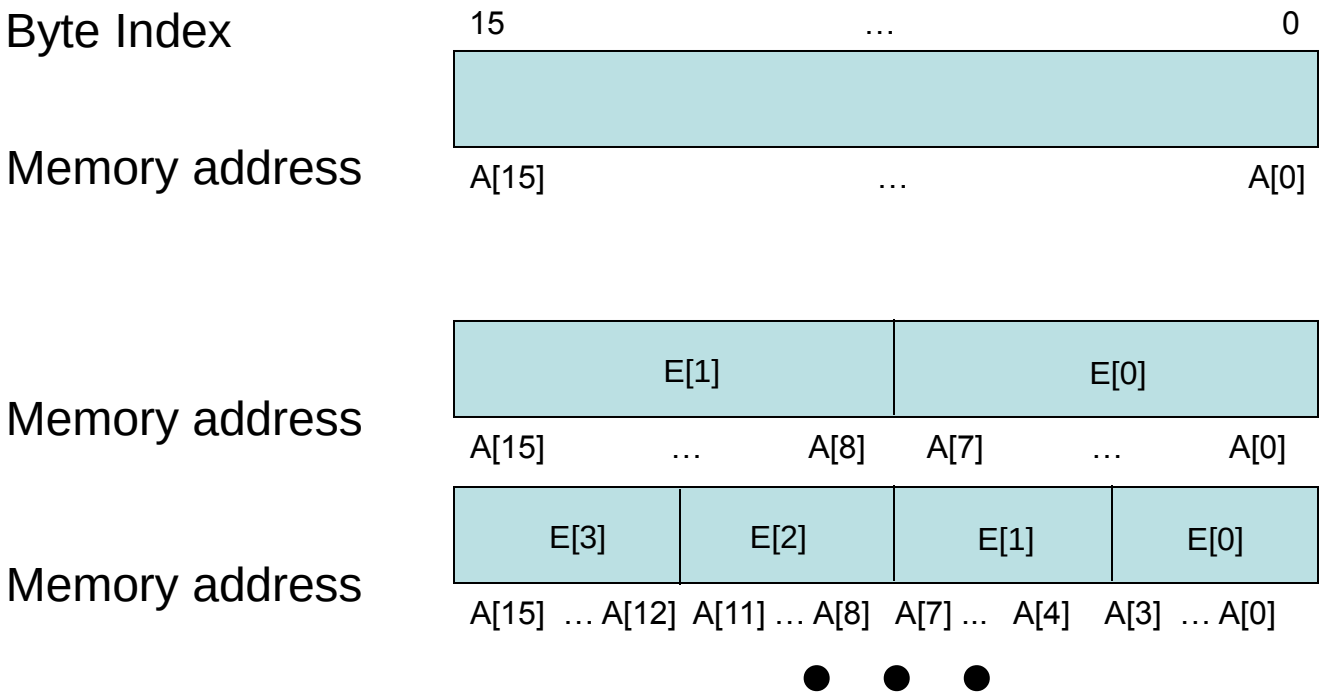
# OpenPOWER Vector SIMD Programming Model

- Transcends traditional hardware-centric SIMD programming models
  - Vector SIMD API describes *what* SIMD processing to perform, *not how*
  - API targets compiler, not hardware primitives
  - API independent of underlying hardware

- Goals: intuitive programming models and application portability
  - from other little-endian environments
    - consistently little-endian API and data layout
  - from big-endian environments (Power BE)
    - simplify porting and sharing of libraries between LE and BE environments
    - exploit large body of numeric middleware developed for Power

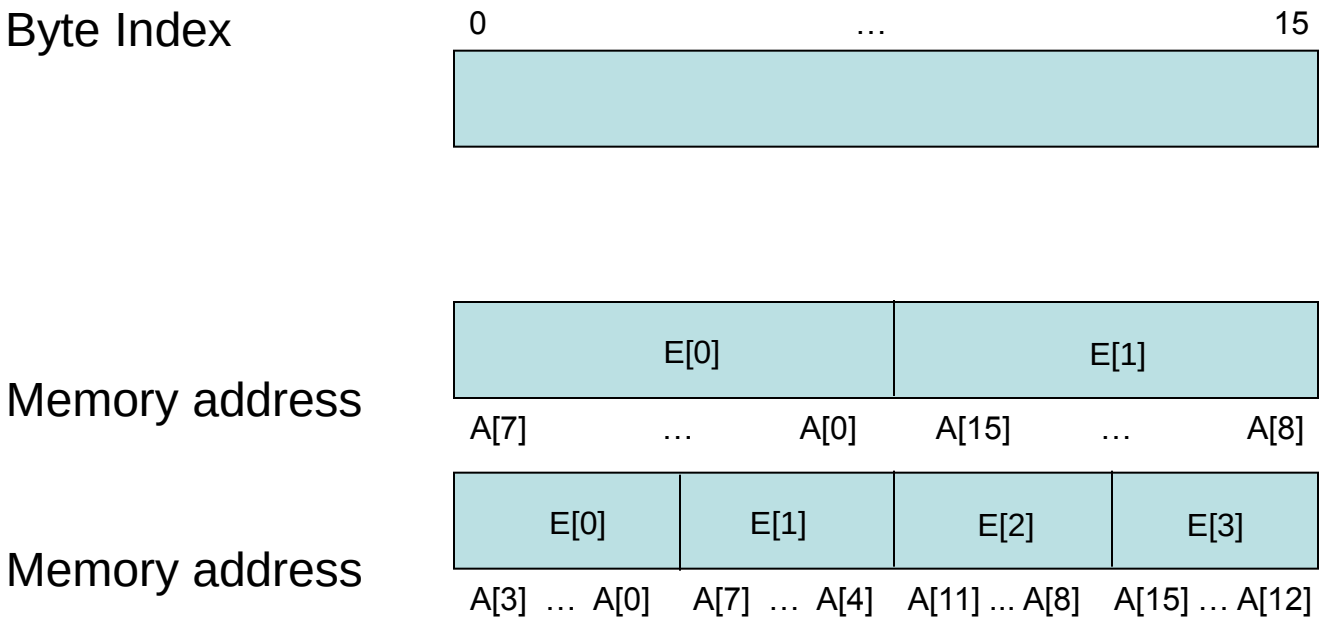
# The “True Little-Endian” Vector Model (Default LE Vector API)

- OpenPOWER Little-Endian Model
  - Focus on programmability – consistent little-endian view
  - Focus on ease of sharing code with other little-endian ecosystems



# The “Big-on-Little-Endian” Vector Model (Optional BE Portability Model)

- “Big-on-Little-Endian” Model ➔ Optional Model for Porting of BE Libraries
  - Focus on porting of Big-Endian code with Big-Endian register layout
  - Focus on ease of sharing code between LE and AIX / BE Linux




● ● ●

# OpenPOWER Little-Endian API Implementation

- Vector API model selected by command line switch
  - Native (default) API is to “True Little Endian”
  - Compiler command line to select “Big-on-Little” (GCC,XL)

- Compiler-based mapping of vector API to hardware primitives
  - True LE ➔ hardware instructions use big-endian register numbering
  - Big-on-Little ➔ bridge BE/LE data layout on memory accesses
    - Power ISA provides support for transparent data adjustment

- Both models translated to common intermediate format
  - Compiler optimizes vector computations

# OpenPOWER available now

- Collaborative innovation already changing industry
  - Major data center stakeholders joining OpenPOWER
- Redefined software stack: Firmware, Hypervisors, OS, Applications
  - little-endian data model for simplified application porting
  - Linux distros available now ➔ 40000 packages ported
- New OpenPOWER environment enables
  - ease-of-use and out-of-box performance
  - exploitation of new Power8 hardware features

# Thank you!

# Special Notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

# Special Notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Watson, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerLinux, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, POWER7+, POWER8, Systems, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: http://www.**ibm.com**/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
AltiVec is a trademark of Freescale Semiconductor, Inc.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.
PowerLinux™ uses the registered trademark Linux® pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the Linux® mark on a world-wide basis.
Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.
NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.