

Article

Introduction to Linux on IBM Power Systems

A brief history of the IBM Power platform and information for Linux developers to help them deploy to Power with confidence

By Paul Clarke

👍 4 ☆



On this page

Overview

[Introduction](#)[Linux](#)[IBM Power](#)[Linux on Power](#)[A fully open system](#)[Deploy with confidence](#)[Architecture differences](#)[Instruction Set Architecture \(ISA\)](#)[Bootting, firmware/BIOS](#)[Hypervisor](#)[/proc and /sys](#)[Packages](#)[Hardware threads](#)[NUMA](#)[A note on processor frequency](#)[A note on benchmarks](#)[Summary](#)

Introduction

The Linux® on the IBM® Power Systems™ ecosystem combines some of the world's best operating systems with one of the world's best processor architecture families: open source with OpenPOWER hardware optimized for performance, reliability, and resiliency. This article provides a very abbreviated history of the platform and information for users to help them deploy with confidence.

Linux

There have been volumes of content written about the history of operating systems, UNIX®, the GNU project, and the Linux kernel; this article will not attempt to repeat that content.

Linux is a fairly young operating system. It started taking shape in the late 1980s. It remained a hobbyist's domain through much of the 1990s, through the *UNIX wars*. It started catching the eye of corporations, including IBM, in the late 1990s. In the 2000s, IBM announced it was investing one billion dollars (US) in Linux development, including helping to port the kernel, the rest of the operating system, and its own software to run well on IBM's servers. Linux is now arguably the most popular operating system on earth (and beyond), running on everything from watches to enormous supercomputers, essential to space exploration and the Fortune 500.

At its heart, Linux started in response to frustration. Developers simply wanted to be able to do what they wanted, how they wanted, and when they wanted. They were certainly skilled enough to do so. However, they were living in a world of closed systems and proprietary, licensed operating systems and software which offered no ability to see the internal workings, nor to change them, share their changes, or collaborate with others in doing so.

The *open source* movement also started long ago, mostly dabbling in tools, games, and the occasional foray into kernels. The GNU project launched an effort to build a complete open source operating system in the mid 1980s, and the combination of the GNU project's work in operating system components and the Linux kernel resulted in a fully open source operating system that allowed every developer to *scratch their own itches*.

Putting all the pieces together tended to be a chore until Linux *distributions* started doing the packaging and simplifying installation so that the Linux operating system could be easily utilized by more than just developers. Slackware, Debian, Red Hat®, and SUSE provided generally useful operating systems for readily available computers.

The predominant processors of the day were produced by Intel® and competitors that produced compatible processors, such as AMD.

IBM Power

Computer programs are essentially a collection of data and a series of instructions for processing that data. Early computers had fairly complicated sets of instructions which nevertheless got the job done. Researchers started looking at a revolutionary approach for processor design: a smaller set of simpler instructions. This came to be known as reduced instruction set computing (RISC). As RISC evolved over the 1980s and 1990s, it had some advantages over traditional complex instruction set computing (CISC) processors, which allowed for simpler circuit design, higher frequency processors, and better performance. RISC processors are not compatible with CISC processors, so they have evolved along parallel and competing paths.

IBM's early RISC products included the IBM RT PC (1986). The IBM RISC processors have continued to mature and compete ever since, with early IBM Power® processors, IBM PowerPC® processors, today's IBM Power9 processors, and the [recently announced tenth-generation Power10 processors](#). These processors have and continue to run mission-critical applications worldwide and support several world-class enterprise-level operating systems including IBM AIX®, IBM i, and Linux, scaling from a single server to the largest supercomputers in the world. IBM Power Systems are renowned for their performance, reliability, and resilience.

Linux on Power

Early in the 2000s, IBM started supporting Linux running on IBM Power processors, including enterprise-class Linux distributions in collaboration with Red Hat, SUSE, and Canonical (Ubuntu).

With IBM's world-class virtualization technologies, IBM Power Systems, can simultaneously run workloads using Linux, IBM AIX, and IBM i operating systems.

10 November 2021

Time to read: 11 minutes

This content also appears in: ▾

Legend ⓘ

Categories

 ▾[Linux on IBM Power](#)[Linux](#)[IBM Power](#)

Interested in generative AI?

[Learn generative AI skills →](#)

IBM Power Systems running Linux consists of IBM's top five entries on the *Top 500* supercomputers in the world, and 9 out of 11 overall.

The combination of Linux-based operating systems and Power processors brings together the open, vibrant, cross-platform nature of the Linux software ecosystem with the powerful, reliable, resilient Power hardware ecosystem.

A fully open system

In the same spirit of openness which launched the Linux kernel and continues to carry it forward, IBM partnered with several leading technology companies to launch the OpenPOWER Foundation in 2013 to share governance of the ecosystem surrounding the Power processor, systems, and software architecture.

With the release of a Power processor core design for free in 2020, under a Creative Commons license, the Linux on the Power platform stack now includes open components from top to bottom:

- Open source enterprise-class Linux-based operating systems
- Open source Linux kernel
- Open source KVM hypervisor
- Open source Skiboot/OPAL firmware
- Open hardware design
- Open ecosystem governance at the OpenPOWER Foundation (now part of the Linux Consortium)

Further, in 2019, the OpenPOWER Foundation with IBM released the Power Instruction Set Architecture (ISA) under an open license, allowing anyone to freely build their own chips in compliance with the Power ISA.

The result is the most open, mass-market, enterprise computing platform in the industry.

Time-tested, powerful, performant, reliable, resilient, and open. What else is there to know?

Deploy with confidence

Deploying a Linux-based software stack on a Power processor-based platform can be done with as much confidence as with any other supported enterprise-class platform with the knowledge that:

- The same enterprise Linux distributions are supported
- The Linux kernel is built using the same source code
- Operating system packages are built with the same source code
- The same toolchain is available and used to build the packages
- The same virtualization technology (KVM) is available
- The same containerization technology is available
- The same system management software infrastructure can be used
- The same databases are available
- Interpreted languages, such as Python, Node.js, Ruby, Java™, JavaScript, PHP, Perl, R, and so on, all run identically without changes

Skills acquired in any of the above areas are perfectly portable from one processor architecture family to any other.

Architecture differences

The Power processor architecture family is a primary architecture family in computing (others include IBM Z®, Intel 64, and ARM AArch64). While there are great similarities among these families, there are also differences.

Instruction Set Architecture (ISA)

The primary difference among families is the Instruction Set Architecture (ISA). Compiled source code (binary code) for one family will not work on a system of another family. Source code which needs to be compiled includes C, C++, and FORTRAN. Fortunately, these languages are well-documented as official standards, and those standards are well-supported by open source compilers such as GCC. So, standards-compliant source code can most often simply be recompiled and relinked.

Code that is written for a specific family, for example assembly, would need to be rewritten. However, assembly code is rarely used.

Bootting, firmware/BIOS

There will be observable differences in the booting process, but the functionality is essentially the same, and the result (a running system) is the same.

IBM Power processor-based systems use OPAL/Skiboot open firmware. Intel 64 uses Extensible Firmware Interface (EFI) BIOS. Observably, they are nowhere the same, but their purpose is to control some low-level system configuration and load and launch the bootloader.

As noted above, OPAL/Skiboot firmware is open source.

Hypervisor

The open source KVM hypervisor technology is available on Power processor-based systems as with other architecture families. The KVM hypervisor on Power processor-based systems supports virtual machines running both little endian and big endian Linux simultaneously.

IBM Power Systems also support the world-class Power Hypervisor (PHYP). PHYP supports virtual machines with Linux (little endian), Linux (big endian), AIX, and IBM i simultaneously.

/proc and /sys

Some files in /proc and /sys will look different than on systems with different processor families due to unique characteristics of the various families. Almost universally, the same basic information is available through more architected means. (In other words, determining the number of processors by parsing /proc/cpuinfo is not recommended, instead favoring something like `getconf _NPROCESSORS_ONLN`, or parsing /sys/devices/system/cpu, which is more consistent among processor families.)

Packages

The base names of packages are the same, but for architecture-specific packages, the architecture is included in the complete package name. For example, the Linux kernel package might be named `kernel-4.18.0.ppc64le.rpm` for a Power processor-based system, and `kernel-4.18.0.x86_64.rpm` for an Intel 64 system. This difference generally only matters when downloading and installing packages manually. Package management systems handle the architecture specifics transparently. On disparate systems, the above package could be installed with the following command:

```
yum install kernel
```

Hardware threads

A Power processor core is robust enough to support more than one or two hardware threads simultaneously. Both 4 (SMT4) and 8 (SMT8) simultaneous hardware threads are common. The threads are "always present", but may be disabled (individually, or uniformly to 2 threads per core (SMT2) or single-threaded (SMT1/ST)). The numbering of the hardware threads (CPUs) by the Linux kernel are core-major. For example, the first core's hardware threads will manifest as CPUs 0 through 3 (SMT4) or 7 (SMT8). In SMT2 mode, the core's hardware threads will still be CPU 0 through 3/7, but only CPUs 0 and 1 will be online.

Other common processor architecture families may number their hardware threads as thread-major, so the first core of *n* cores would have CPU 0 and CPU *n*.

Given that Power processor-based systems can have a large number of processor sockets, and many cores per processor, and many threads per core, there are potentially thousands of CPUs available to applications. An IBM Power E1080 server supports up to 16 sockets. At 15 cores per processor and 8 hardware threads per core, that works out to $16 \times 15 \times 8 = 1920$ CPUs!

Having a system with robust processors can be a significant advantage under some software licensing models that count cores (240 in the example above) instead of CPUs (1920).

NUMA

While operating systems present memory to the running applications as a unified space, modern large systems have memory organized in a hierarchy, by core, chip, module, socket, node, and chassis/central processor complex (CPC) for example. A memory reference from a core may have to reach far away, logically, to be resolved, resulting in much higher latencies. Different memory addresses having different latencies is called nonuniform memory access (NUMA).

Large, complicated, multi-task applications may need to take extra care in the placement/binding of their tasks and memory such that memory that is frequently accessed by each task is in a location with low relative latency with respect to that task, and that the task be prevented from migrating to a different processor location.

A note on processor frequency

In the discussions above regarding RISC versus CISC, and the various processor architecture families, it should be obvious that different ISAs means different instructions performing different tasks. Instructions from one ISA might be generally more complex (or more simple) than those from another ISA. Because of this, it is not appropriate to draw firm conclusions about relative performance of processors from different architectural families based solely on processor frequency or a metric like instructions per cycle. Benchmarks of actual code on actual systems is much more meaningful.

A note on benchmarks

The best benchmarks are those that most closely mimic what the system will be doing in production. So-called *microbenchmarks* which test a tiny subset of code (for example, a single thread performing a single library call a million times) are generally poor indicators of production performance. Power processor-based systems are built for bandwidth, throughput, capacity, reliability, and resiliency, and microbenchmarks often fail to push those boundaries.

Summary

Linux on IBM Power Systems offers a radically open solution stack from the hardware up through the operating system, optimized for performance, reliability, and resiliency.

Take the next step

Join the [Power Developer eXchange Community \(PDeX\)](#). PDeX is a place for anyone interested in developing open source apps on IBM Power. Whether you're new to Power or a seasoned expert, we invite you to join and begin exchanging ideas, sharing experiences, and collaborating with other members today!



IBM Developer
About
FAQ
Third-party notice

Follow Us
X
LinkedIn
YouTube

Explore
Open Source @ IBM
IBM API Hub

[Career Opportunities](#) [Privacy](#) [Terms of use](#) [Accessibility](#) [Cookie preferences](#) [Sitemap](#)