

JANUARY / FEBRUARY 2006
WWW.EOSJ.COM



ENTERPRISE
**Open
Source**
JOURNAL

Focusing on Open Source Strategies in the Enterprise

How to Evaluate
Community Support
for an Open Source
Content Management
System

The "B" in LAMP:
How Berkeley DB
Helps LAMP Shine

The Buzz About
Enterprise
Service Bus (ESB)

10 Popular Open
Source Content
Management
Systems

DEMYSTIFYING

myths

SURROUNDING
OPEN SOURCE



THE “B” IN LAMP

How Berkeley DB Helps LAMP Shine

BY BILL WEINBERG

LAMP—the Linux operating system, the Apache Web server, the MySQL database, and Perl/Python/PHP scripting languages—together form an open source software stack that powers much of the Internet and comprises the base platform for many enterprise information systems. LAMP’s widespread use showcases many benefits of Open Source Software (OSS)—low-cost, flexibility, customizability, reliability, and excellent performance. OSS in the LAMP stack leverages the vibrant development community and contributions from the larger OSS ecosystem.

Spelling out the components and the benefits of developing with and deploying LAMP-based applications can give the impression that LAMP shines alone. However, Linux, Apache, MySQL and Perl/PHP/Python aren’t the only software technologies that illuminate this strategic stack. Like most open source (and indeed much proprietary software), the LAMP stack and each of its components build on other salient software. So, no one should be surprised that each component in LAMP relies on the open source Berkeley DB database for various types of data management.

If you aren’t familiar with Berkeley DB, you should know that it provides a ubiquitous, high-performance, non-relational database. Berkeley DB has a worldwide developer community and commercial development and support from Sleepycat Software of Emeryville, CA. LAMP depends on Berkeley DB, as do dozens of other OSS projects (see sidebar) and high-profile, Web-based companies such as Amazon, AOL, Google, and Yahoo.

This article examines the use of Berkeley DB in the LAMP stack and how it provides fast, local persistence with low administration overhead; it offers criteria for evaluating database development and deployment.

Lighting the LAMP

Before we attempt to solve the orthographic conundrum of finding “B” in “LAMP,” let’s briefly examine how the key elements of the LAMP stack fit together to support Web application and enterprise system development (see Figure 1).

Modern Web-based and other enterprise applications present a range of common platform requirements. Beyond “core value” needs such as reliability and performance, they share a key set of technical and architectural underpinnings: >

System Services: Linux

Applications of any stature or complexity look to an operating system for scheduling, networking, low-level I/O, and security. In Linux, they find a robust, standards-compliant and open platform base, with high-throughput TCP/IP networking and integrated security mechanisms. For portability and simplicity, applications seldom look directly to the OS for these core capabilities; rather, they “touch” the public interfaces provided by libraries such as GNU glibc or “wrappers” around other system functions.

Web Services: Apache

Many modern applications manifest themselves primarily as back-ends for Web-sites. They provide the transaction support for e-commerce sites such as Amazon and financial services sites such as Charles Schwab, massive data stores and search engines behind Google and Yahoo, and the business rules and contact management underlying Customer Relationship Management (CRM) systems such as Salesforce.com and SugarCRM.

Other enterprise applications aren't Web-based per se, but leverage HTTP and other protocols to present management and configuration interfaces to systems whose focus lies elsewhere. Examples include secure configuration of massive telecommunications systems, from metropolitan area wireless switching to provisioning and setup of individual mobile or Internet Service Provider (ISP) user accounts. Similarly, many embedded applications let users set up wireless routers, intelligent printers, and multi-media systems with Web-based configuration screens.

These and other applications rely on Apache, the open source Web server that, according to Netcraft.com, runs almost 70 percent of all Websites worldwide. Apache serves up informative and dynamic Web pages, quickly and securely, and scales from one to thousands of users and gigabytes of traffic daily. Applications build on Apache through a combination of static Web content, dynamic pages created from data stores and presented with Java and JavaScript, and through Common Gateway Interfaces (CGIs), written primarily in Perl, PHP, and Python, that manage incoming data.

Data Store and Database: MySQL

Simple Web-based applications and application management interfaces typically

evolve from using ad hoc data stores (in raw text or XML, parsed by Perl, PHP or Python scripts) to relying on off-the-shelf databases, starting with flat-file systems and as requirements dictate, growing to using more robust database systems such as Berkeley DB. The most complex applications demand full relational databases, and the leading open source database is MySQL (and also open PostgreSQL and proprietary systems such as Oracle that run on Linux).

Application Glue: Perl, PHP, and Python

Linux-based systems inherit design and programming philosophy from 35 years of Unix development. A key tenet of both Unix and Linux application design is to limit the complexity of each system component, and create new functionality by connecting and reusing existing pieces of software. For 100 percent native Unix and Linux applications, the “glue” that holds components together was traditionally provided by shell scripting (C and Bourne shells, Tcl, and more recently, bash). Web-based application developers do leverage Linux shell programming, but more frequently look to self-contained scripting environments such as Perl, PHP and Python to bind together Web, data store and application components, and to present OS services to them.

Highlighting the “B”

So where is the “B” in LAMP? Each of the four technologies that support Web application and enterprise system development builds on capabilities of Berkeley DB in similar and also unique ways.

The “B” in Linux: Since the Linux kernel sits strategically at the base of many applications platforms, including LAMP, it must be

extremely robust and reliable. While the entire Linux source tree may top 2 million lines of code, design principles still dictate that the kernel should strive for simplicity and reuse code as much as possible for optimal run-time performance and size. Because it forms the base of the platform, the Linux kernel must also be self-contained; it cannot easily rely on layers above for basic functions while it supplies those layers with system services.

All operating systems must maintain tabular data, sometimes in massive scale. The Linux kernel manages lists of running processes/threads, allocated memory pages, virtual address mappings, synchronization objects, virtual file systems and file “handles,” open network interfaces, installed device drivers, users, user groups and passwords, and myriad other constructs.

Often, the Linux kernel project maintains kernel-based libraries of functions for internal, self-sufficient use (e.g., kmalloc() and kprintf()), and indeed, many kernel data management tasks also fall to special-purpose kernel library functions. For the time and strategic tasks of user and network authentication, the Linux kernel embeds a version of Berkeley DB to ensure sprightly response for user login and secure network operations.

The “B” in Apache: The main job of a Web server is to “listen” for requests coming in over a network interface (socket) and to respond appropriately. Apache listens for HTTP “GET,” “HEAD,” and “PUT” requests, handing back Web page text, graphics and other content and Web page metadata, and uploading forms content and other input data used to drive CGI scripts or to hand off to back-end applications. These and other Web transactions can occur “in the open,”

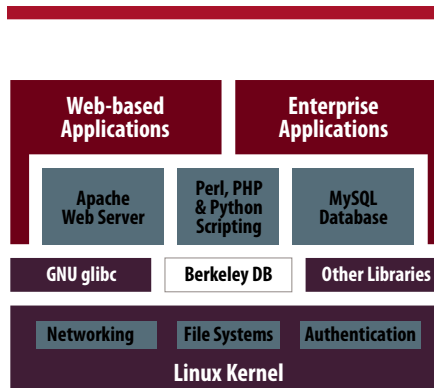


Figure 1: Key Elements of the LAMP Stack

but are increasingly subject to a range of security measures to ensure both system integrity and user privacy.

Like the Linux kernel, Apache relies on and embeds several versions of Berkeley DB to support authentication. Basic HTTP authentication is supported with a standard version of the database; the Apache Directory Server embeds a Java-based Lightweight Directory Access Protocol (LDAP) server, which depends on Berkeley DB Java Edition; Apache Jakarta (Java Caching System) offers distributed server-side caching for Java applications and also uses Berkeley DB Java Edition for managing disk storage.

The “B” in MySQL: Relational databases and Relational Database Management Systems (RDBMSes) exist to support complex applications that store data and highlight and build on relationships among data items. An RDBMS needs to work at several levels, starting with solving the same indexing and searching problems as a flat database, and progressing to address the complexity of tying together tables with relational algebra.

While RDBMS systems differentiate themselves with their relational capabilities, they still must do a good job at table management and indexing. MySQL, the popular open source SQL database system, relies on Berkeley DB to let users create and manage the tables that compose the RDBMS. The combination of the MySQL query engine and Berkeley DB storage services gives MySQL users fast, reliable, transaction-protected data management services driven by the power and flexibility of the standards-based SQL query language.

The “B” Perl/PHP/Python: Whereas the Linux kernel, the Apache server and the MySQL RDBMS embed and build upon Berkeley DB for their core functionality, programming tools such as the Perl, PHP and Python languages strive to give developers clean, orthogonal and easy-to-program interfaces to Berkeley DB management functions. By providing libraries for and wrappers around Berkeley DB Application Program Interfaces (APIs), Perl, PHP and Python let developers use their most comfortable idiom to create, access and manage instances of Berkeley DB database tables.

For both Web-based and other enterprise applications, Perl, PHP and Python

Database Terms & Types

Databases have evolved over the history of the IT industry from supporting linear tables of data to supporting cross-reference among that data, to emphasizing relationships among tables, to supporting object-oriented data and programming models, to handling networked distribution of data stores.

The following definitions should help readers understand how Berkeley DB fits into applications and other database systems:

- **Database:** a well-defined set of data or the software used to organize, access, and maintain it.
- **Flat database:** a collection of data items, organized into fixed-format rows (records), with separate columns to represent different types of data (e.g., name, address, birth date). Spreadsheets are a form of flat databases (or programs to manipulate them). Flat database software adds value in how fast it supports access to large data tables, and in how secure and robust its data storage (persistence) methods are. Examples include open source software such as Berkeley DB and proprietary programs such as FileMaker.
- **Relational database:** flat databases manage single large tables of fixed-form data, and relational databases help users create, manage and manipulate data stores with multiple tables of different kinds of data (record format can differ table to table), and create relationships among those tables and the data within them. Relational databases are, by definition, more complex than flat database systems, with different performance characteristics and more complex platform requirements. Examples of open source databases include MySQL and PostgreSQL; proprietary relational databases form the core businesses of companies such as Oracle and Sybase.
- **Embedded database:** either a database manager contained within another program (as with Berkeley DB and MySQL) or a database manager optimized for embedded applications (e.g., mobile phones, telecommunications infrastructure, industrial control systems, etc.). Berkeley DB is both!
- **Index:** a pointer to a row in a flat database or a relational database table. It can be an arithmetic value that represents the nth entry in the database or an instance of the data type that constitutes the table column (a unique key; e.g., your name in a table of readers of this magazine).
- **Key:** an index to a table whose value is of a type represented in one column of that table. Keys can be common or unique and are used to create relations among tables in relational databases.
- **SQL:** a standard set of commands and APIs used for console-based and programmatic interaction with (primarily) relational databases.

-B.W.

	AD HOC	FLAT DB	RDBMS
Operations			
Indexing	Varies with size	Fastest	Fast
Querying	Varies with size	Fastest (single table)	Fast across tables
Table Definition	Variable	Usually fixed	Flexible
Attributes			
DB Size (Recommended)	Small	Small/medium/large	Medium/large
DBM Size Standards-based	Smallest No	Small/medium Often	Medium/large Usually (SQL)
Licensing			
Open Source	It's yours!	Berkeley DB et al.	MySQL, PostgreSQL
Commercial Support	No	Yes (Sleepycat et al.)	Yes (MySQL et al.)
Cost	Free / maintenance	Low	Medium to high
Packaging	None	Stand-alone or solution	Usually solution
Applications			
Enterprise	Application-specific	Infrastructure	Core business
Desktop	Application-specific	Core productivity	Local and remote
Embedded	Extremely common	Memory-resident or Flash-based	Memory-resident or with RDBMS client

Figure 2: Database Selection Criteria

bindings to Berkeley DB APIs let developers both access and extend data stores directly associated with those applications, and set up well-formed databases to support infrastructure “glue” requirements so frequently implemented into those scripting languages.

Beyond LAMP

While guiding the reader on choice of database manager type and particulars is beyond the LAMP-centric scope of this article, it’s instructive to review the considerations for choosing a database for different types of applications (see Figure 2).

While there’s no one-size-fits-all database and DBM technology, a flat database such as Berkeley DB offers a good combination of cost, performance and scalability, with abundant applications across enterprise, desktop, and embedded design domains.

Votes for Berkeley DB

The Berkeley DB project dates back over a decade to improved hashing code for ATT and Berkeley Unix, and for the GNU project. Today, Berkeley DB enjoys an active life and thriving community as an open source project. For more information, visit <http://freshmeat.net/projects/>

berkeleydb/ and www.sleepycat.com.

Sleepycat Software offers commercial packaging and support for Berkeley DB, and invests substantial resources worldwide in project development and code base upkeep. While Berkeley DB use spans decades and time zones, applications and business models, Sleepycat customers, in particular, provide insight into the quality and performance of the DBM.

For example, Sleepycat states that: “A major Internet company estimated they could save between \$2 and \$5 million in hardware procurement costs by using Berkeley DB instead of a traditional RDBMS such as Sybase and Oracle. For the sake of comparison, a 64 CPU server running Sybase or Oracle can process 6,000 transactions per second, while a 2 CPU server running Berkeley DB can process 10,000 transactions per second. [This works out to 26 times more transactions per CPU.]”

Similarly, for an embedded application, Ray Van Tassle, senior staff engineer at Motorola, notes that: “Berkeley DB was 20 times faster than other databases. It has the operational speed of a main memory database, the startup and shutdown speed of a disk-resident database, and doesn’t

have the overhead of client/server, inter-process communication.”

For Java-based enterprise applications, Eric Jain, an engineer at the Swiss Institute of Bioinformatics, adds, “With Berkeley DB Java Edition, we have a simpler setup, a 3x increase in data import speed, a 5x increase in performance and a 10x decrease in disk storage requirements.”

Conclusion

There’s no reason to have your eyes checked or to consult a speech pathologist. There really is a “B” in LAMP and it’s the B in Berkeley. The Berkeley DB forms a key component of the increasingly ubiquitous LAMP stack and also LAMP’s Solaris (SAMP), Windows (WAMP and WIMP), and Macintosh (MAMP) incarnations. It also empowers dozens of other open source software and thousands of commercial applications. So don’t be surprised to find a “B” in your next application architecture. ●

Bill Weinberg brings more than 18 years of open systems, embedded, and other IT experience to his role as open source architecture specialist and Linux evangelist at the Open Source Development Labs, where he participates in OSDL initiatives for Carrier-Grade, Data Center, and Desktop Linux.
e-Mail: bweinberg@osdl.org