SPONSORED BY

IBM

GEEK GUIDE

Why Innovative App Developers Love High-Speed OSDBMS

LINUX JOURNAL

Table of Contents

Ted Schmidt is a consultant who specializes in marketing and e-commerce solutions for the manufacturing sector. Ted has worked in project and product management since before the agile movement began in 2001, and he has managed project and product delivery for consumer goods, medical devices, electronics and telecommunication manufacturers for more than 20 years. When he is not immersed in product development, Ted writes novels and runs a small graphic design practice at http://FloatingOrange.com. Ted has spoken at PMI conferences, and he blogs at http://FloatingOrangeDesign.Tumblr.com and on his website at http://FloatingOrange.com.

GEEK GUIDES:
Mission-critical information for the most technical people on the planet.

About the Sponsor

IBM

IBM is a globally integrated technology and consulting company headquartered in Armonk, New York. With operations in more than 170 countries, IBM attracts and retains some of the world's most talented people to help solve problems and provide an edge for businesses, governments and non-profits.

Innovation is at the core of IBM's strategy. The company has reinvented itself through multiple technology eras and economic cycles, creating differentiating value for its clients. Today, as the IT industry is fundamentally changing at an unprecedented pace, IBM is much more than a "hardware, software, services" company. IBM is now emerging as a cognitive solutions and cloud platform company.

Cognitive solutions powered by the cloud are the key to clients' digital transformation. This transformation requires breakthroughs at every level of the enterprise IT foundation, from processors and computer design to storage, networking and the integration layer. IBM Power Systems, built with open technologies and designed for mission-critical applications, offers the infrastructure that is designed for cognitive workloads.

Why Innovative App Developers Love High-Speed OSDBMS

TED SCHMIDT

Introduction

As any developer of social, mobile or IoT applications can attest, legacy relational database models aren't serving all our needs anymore. There's nothing wrong with traditional database management systems—they simply weren't designed to address the variety and volume of data gushing through today's digital world, not to mention the demand for speed of processing all that data in order to deliver the useful, data-driven features and functionality all of us increasingly expect

of practically everything. Fortunately, a whole new world of Open-Source Database Management Systems (OSDBMS) has been built precisely to handle the diversity and complexity of today's data and to be able to store, analyze and act on it at the speed required to make it valuable in ways like never before.

In this guide, I take a look at some of the available OSDBMS and the solutions they offer to problems we face in developing innovative apps for new data from sources like social, mobile and IoT. Although discussing *all* the OSDBMS players is beyond the scope of this guide, I offer a balanced look at the lead players in each of the major categories, including open-source SQL, NoSQL (including graph, document and key-value stores), and even the in-memory, GPU-accelerated products available today.

I start by discussing some key challenges of the new application landscape, including the Big Data that new apps create and consume. I give an overview of the OSDBMS landscape, followed by a deeper dive into some of the major OSDBMS offerings. Finally, I conclude with a look at the best technology platform available to make these modern DBMS perform at peak levels, solving challenges and breaking new records of speed, throughput and scale to serve today's most innovative apps.

Challenges Facing Innovative App Developers

We remain abuzz around challenges associated with "Big Data" and its evolution. Although Big Data may have different requirements for different organizations

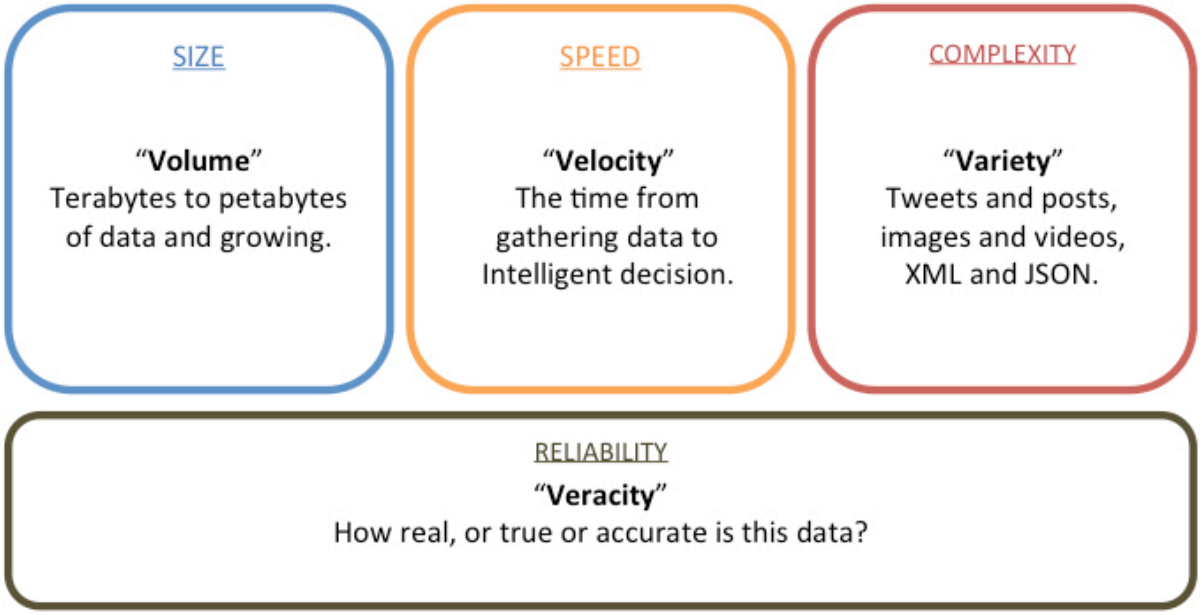

FIGURE 1. Four Dimensions of Big Data

and industries, when I talk about Big Data here, I'm specifically referring to it in terms of the issues faced when designing and building innovative apps in a world that demands instant access to lots of data. I'm considering the external sources of Big Data and the problems associated with capturing, storing, analyzing and visualizing this data. For an app developer focused on providing innovative, data-driven solutions, four dimensions of Big Data are of concern: size, speed, complexity and reliability. In this guide, I focus mostly on the aspects of speed and complexity, and the value of OSDBMS and NoSQL solutions to address them.

The dimension of speed, or "velocity", refers to how fast data is created, stored, processed, retrieved and so on, and ultimately made usable for analysis. More and more,

organizations demand that data be processed in real time and streamed directly into the decision-making processes of a business. Think about traffic-control systems. Although the demands for speed will continue to grow, our ability to keep pace with that demand is hampered by a couple things.

Obviously, latency—the difference between when data is collected and when it's available for use—is going to impact speed. In addition to latency is the looming spectre of Moore's Law. Moore's Law states that overall computer processing power should double every two years. Although that's worked for a while, common wisdom now says there is a physical limit to what silicon chips can provide. It is physics, and it directly impacts the ability of apps to continue to get the speed gains they need to process growing amounts of increasingly complex data in real time.

What all this means is that, as developers, we're more challenged now because we can't simply rely on Moore's Law as the answer to demands for more speed. We have to be more innovative, and that means looking at the benefits of open-source solutions, rather than continuing to rely on traditional databases, to handle the huge volumes of data we're now working with and from which we're striving to deliver useful features, functionality and insights as quickly as possible.

What Open Source Brings to the Game

There are dozens of open-source DBMS solutions. I look at five here that offer particular benefits when it comes to innovative new app development: MongoDB, Redis, Neo4j, PostgreSQL and Kinetica. (Full disclosure: although

Kinetica is not open source, it is part of an open technology ecosystem through its membership and participation in the OpenPOWER Foundation. As such, and because of its particular advantages in real-time processing for large, streaming data sets, I include it in this discussion. The OpenPOWER Foundation, which I talk about later in this guide, is an open technical membership organization that basically seeks to respond to the limits of Moore's Law by allowing member companies to customize POWER CPUs in new and innovative ways in order to squeeze out every ounce of speed and power possible. I highly recommend learning more about the OpenPOWER Foundation at https://openpowerfoundation.org, but suffice it to say here that this is where innovation is happening.)

Moving to an OSDBMS has some tangible benefits, and as the management toolsets continue to mature to enterprise level, former limitations are falling away. Lower cost associated with being freed from proprietary software, and even proprietary hardware, is the most obvious benefit of open source in general. But, as you will see when I cover the specific use cases each of these OSDBMS solutions fits best, speed, flexibility and intelligent decision-making are king. That's not to dismiss the advantage open-source brings to the speed of evolution and innovation. Proprietary solutions by nature evolve more slowly to changing environment demands. Open-source solutions, however, benefit from the input of a broad community of many voices focused on solving real-world problems. As such, open-source solutions evolve and innovate faster than any proprietary solution is capable of doing.

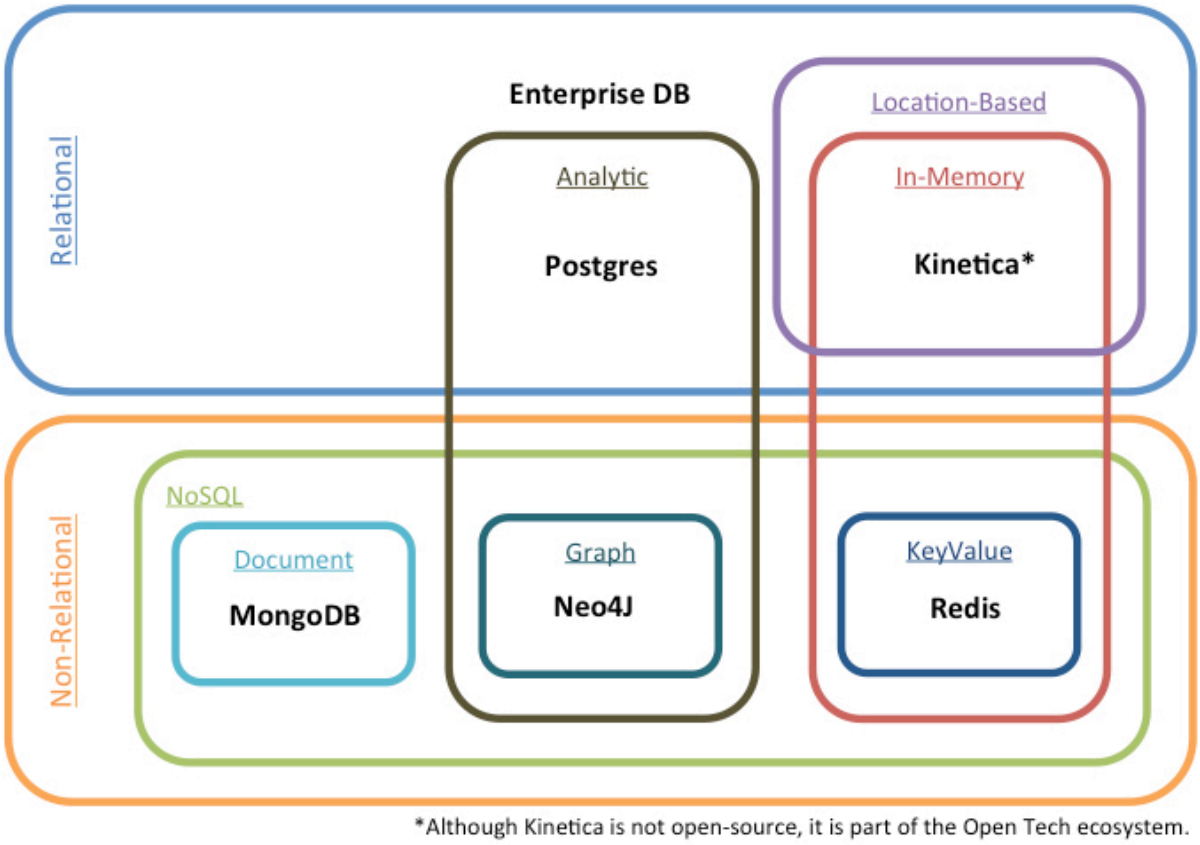

FIGURE 2. OSDBMS Landscape

Let's consider the two main categories of the OSDBMS landscape: non-relational (which includes MongoDB, Redis and Neo4j) and relational (which includes EDB Postgres and Kinetica).

On the non-relational side of the landscape is Redis, a NoSQL key-value database that's particularly useful in the gaming industry (among many others) where high-speed, simplistic yet elegant data ops are king. Next, Neo4j is a NoSQL graph database that's especially effective at storing the relationships between data points. Finally, MongoDB is a NoSQL document database, making it great as a

general-purpose database but particularly useful for its lack of a schema, meaning you can store all kinds of different data in a highly flexible manner.

For the relational category, Kinetica is not open source but merits recognition for the pure speed it brings to the table. A relational database that's also location-based, Kinetica is an in-memory, GPU-accelerated database. The key to its record-breaking speed in processing large, streaming data lies in that last bit about being GPU-accelerated (more on that later). Last but not least, the other relational database I review here, EDB Postgres Advanced Server, is really EDB's enterprise-grade packaging of PostgreSQL, a product of open-source development by a community in which it actively participates and religiously supports. It's also great for analytics.

On the non-relational side of the landscape is another in-memory database: Redis. Redis is a NoSQL, key-value database that is particularly useful in the gaming industry. Neo4j is yet another NoSQL database that, as a graph database, is especially effective at storing the relationships between data points. Finally, MongoDB is also a NoSQL database, but it's a document database, which means that although it's a good general-purpose database, its real benefit is derived from its lack of a schema. Without a schema, you can store all kinds of different data, like rise in temperature or rotations per second.

Keep in mind that although the benefits of each OSDBMS, often overlapping, are ultimately based on your specific use case, all of these databases share a similar benefit of being part of an open technology development

model. Not only that, but on the right infrastructure, each OSDBMS provides the desperately needed bonus of break-through speed for real innovation in data-intensive app development.

MongoDB Document Store DBMS

MongoDB is an open-source database with a document-oriented data model. It stores data in Binary JSON (BSON), which extends JSON to include other data types like int, long, date, floating point and so on. These BSON documents make it much faster and easier to model data in the application to the data in the database, because the BSON documents are aligned to the programming language object structure. Every document contains multiple fields, and each field contains a value of a specific data type, like sub-documents or arrays. Documents that are alike in structure are then grouped together into collections. In an RDBMS, collections would be tables, documents would be rows, and fields would be like columns.

A document-oriented data model has no schemas. So, unlike an RDBMS, which stores NULL values in empty fields, in MongoDB, if there's no data, there's no field to store the data. This means you don't need to worry about changes to an existing schema when you're developing, which makes you much more agile in your delivery of constantly changing business requirements. This opens the door to innovation, because it makes it much easier to evolve applications.

MongoDB's document-oriented model also reduces the need to create joins, because you don't break normalized

MongoDB is especially great when you need to deploy web apps built on JavaScript quickly, make use of a lot of real-time counters or store lots of images.

documents into smaller tables. With fewer joins comes big improvement in scalability and speed. The nice thing about MongoDB, as opposed to other NoSQL databases, is that you still can use joins if you want to combine data from multiple collections.

MongoDB also offers automatic sharding capabilities and support for geo-spatial applications, making it ideal for the type of applications I'm discussing here. Compared to RDBMS partitioning, which is complicated by multiple tables and joins, sharding with MongoDB is done by partitioning the key space, because the key is the document ID, and the document is the value in the key-value document stores. Actually, all NoSQL databases have some form of sharding or partitioning that reduces latency and improves scalability.

Using auto-sharding and BSON documents, MongoDB provides a high-speed and flexible database that enables a more agile and responsive approach to changing business requirements. MongoDB is especially great when you need to deploy web apps built on JavaScript quickly, make use of a lot of real-time counters or store lots of images. It's incredibly fast when used to meet the real-time querying and reporting needs of IoT. And with geo-spatial support,

it's ideal for use in applications when knowing where the user is, or showing the user where to go, is important.

Developers also will appreciate the on-demand training MongoDB offers. MongoDB provides development support that is project-based, versus server-based, which makes it a huge help to both developers and ops managers.

EDB Postgres Advanced Server

EDB Postgres is actually an expanded version of the open-source PostgreSQL relational database, distributed by Enterprise DB. Although EDB Postgres used to be a release behind PostgreSQL, it works really hard at being an active and integral part of the PostgreSQL community.

Speed and scalability are the key benefits PostgreSQL brings to the table, and EDB complements that with a rich set of enterprise-grade tools. Having access to the PostgreSQL community ecosystem is a clear benefit, but other benefits stem from this high-performance and scalable database as well.

PostgreSQL supports multiple data types, including user-defined data types (like XML), table collections and Varrays. It supports text data, indexing and searching, and it allows read/write operations to execute without blocking by employing multi-version concurrency control. Finally, stored procedures can be written in languages like C/C++, Java, JavaScript, Python, Perl and Ruby, which provides plenty of freedom and flexibility.

EDB Postgres also offers enterprise-class security features, including expanded password profiles. Via the open-source PostGIS plugin, EDB Postgres can be implemented as a

back-end spatial database for geographic information systems (GISes). It also comes with a GUI tool for creating and debugging triggers and stored procedures.

Additionally, EDB Postgres provides vertical scaling optimization and expanded scalability improvements for locking subsystems, which improves performance. It makes integration across different databases (supporting relational, document and key-value databases) possible, which allows you to combine unstructured, structured and transactional data together. You also can use it for read-only apps where high speed is critical. It allows DBAs to prioritize both I/O and CPU consumption across processes selectively, and it's compatible with Oracle.

EDB offers a Postgres Developer Subscription that provides direct access to Postgres expertise, technical videos, tons of documentation and an incredibly strong community. EDB also provides an enterprise-class tool suite to relieve the usual burdens associated with migration, integration and management.

It's also worth noting that EDB Postgres is sold as a subscription DBMS, which includes all upgrades, maintenance and support, in addition to the software.

The bottom line is that EDB Postgres Advanced Server is a great modern relational database solution, because it's secure, scalable, flexible and fast, especially when running on an optimized server architecture (more on that later). It delivers everything you expect from an enterprise-grade RDBMS, without the extra-hefty licensing fees associated with other vendors and with the innovation you appreciate from open source.

Neo4j Graph Database

Although it may seem like it at first, there's really nothing magical about graph databases. A graph is composed of two basic elements: nodes and relationships. Every node represents a piece of data—a thing, an entity. Every relationship represents how two nodes are, well, related to each other. Social-networking sites where users follow each other, like Facebook or Tumblr, are classic examples of this idea. The users are the nodes, and the "following" is the relationship between the nodes.

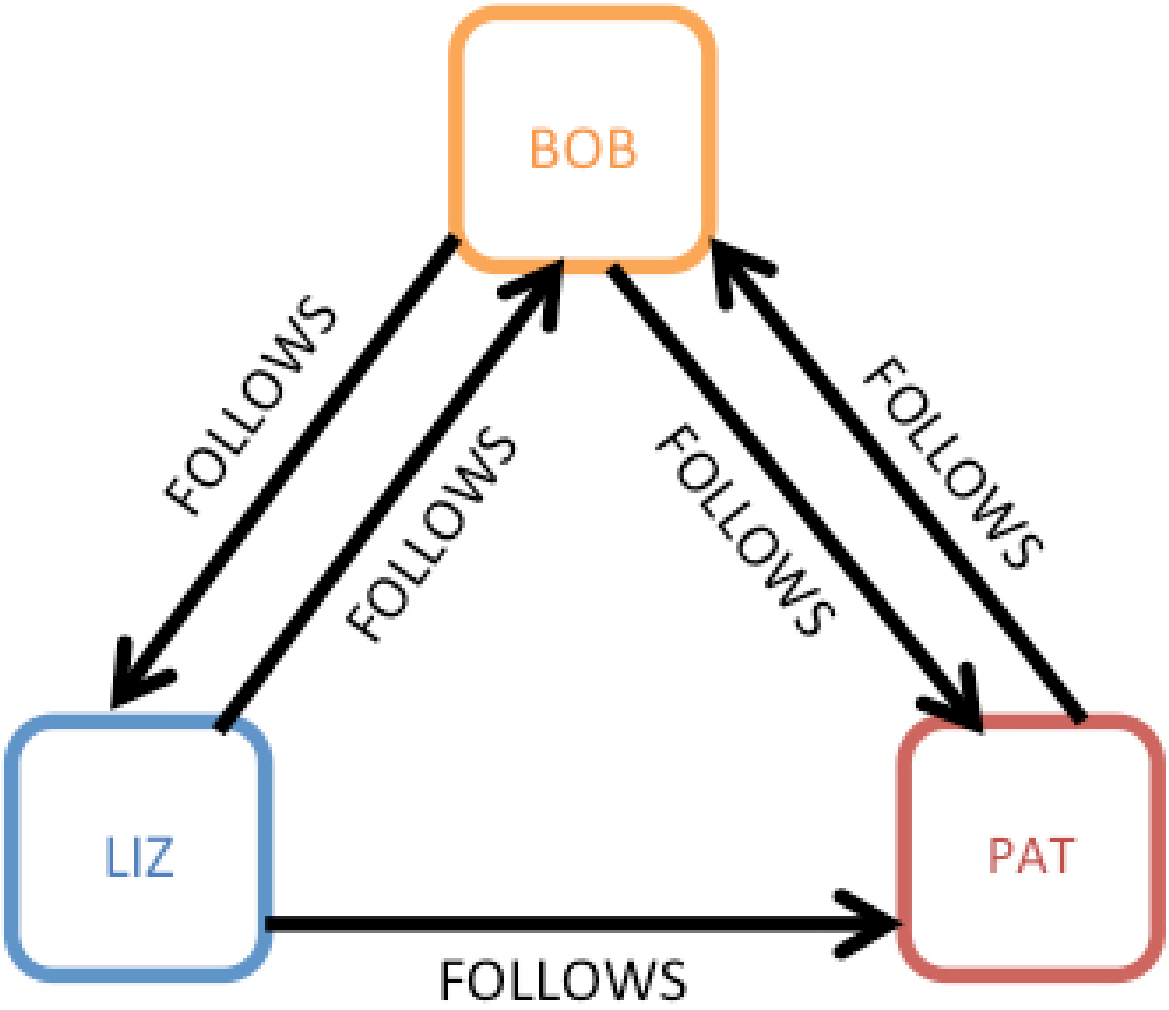

FIGURE 3. Nodes and Relationships

In a graph database, the relationships take first priority, which means data models are simpler and more expressive, and you don't have to worry about things like foreign keys. It also means you never can have a relationship without two nodes, and you can't delete any node without also deleting its relationship(s).

Graph databases have two key characteristics that are important to consider in understanding the benefit a graph database delivers to app development. The first is the difference between native and non-native graph storage. Native graph databases like Neo4j are specifically designed to store and manage graphs, as opposed to adapting relational or object-oriented databases to be graph-like. Non-native graph databases use relational or object-oriented databases for data storage. So, when data volume and query complexity increase, a non-native graph database ends up being a lot more latent.

Neo4j's second beneficial characteristic is its graph-processing engine. In native graph processing, connected nodes point directly to each other. This is called index-free adjacency, and it's the most efficient method for processing data in a graph database. Neo4j's native graph-processing engine provides constant, real-time performance because it avoids expensive index lookups that non-native database have to perform.

These characteristics are great for identity and access management applications, where you need high-speed tracking of users and authorizations, or real-time recommendation engines that drive many e-commerce product and personalization apps. And, as I implied earlier,

those characteristics are indispensable when you need real-time analysis of social app data.

Neo4j is particularly agile thanks to its adaptable data model—you can respond to emerging business needs with data model changes without worrying about impacting existing functionality. Neo4j also is built for speed. Because of the explicit relationships between nodes, Neo4j avoids the inevitable ensuing slowness when the dataset grows.

Neo4j also provides excellent on-line developer support, including a hefty document library, knowledge base, access to sandbox environments and, like the other OSDBMS I've described, a strong community support system.

In-Memory Database Systems

In-memory database systems (IMDBS), such as Redis and Kinetica, store data in main memory, in contrast to traditional database systems that are designed to store data on persistent media. Although you technically could put a traditional database into RAM, you'd still be saddled with the overhead of a system designed for disk storage. Precisely because an IMDBS stores data in memory, thereby avoiding the overhead of I/O operations and caching, it is incrementally faster than a traditional DBMS. IMDBS also have much lower memory and CPU requirements because they have such a simple design.

Applications that need very fast access to, and manipulation of, data are great candidates for IMDBS. Real-time embedded systems, financial market applications, e-commerce and social applications are excellent candidates for IMDBS, because they can gain real benefits from their

It's not uncommon for an IMDBS to grow beyond the terabyte size range while maintaining all the performance advantages over traditional DBMS solutions.

speed. And not only is IMDBS fast, it also scales very well. It's not uncommon for an IMDBS to grow beyond the terabyte size range while maintaining all the performance advantages over traditional DBMS solutions.

Redis A key-value database, or store, is a database designed for storing, retrieving and managing associative arrays. An associative array is a simple data model where every key is associated with a single value in a collection—a relationship referred to as a key-value pair. An arbitrary string, like a filename or hash or URI, represents the key in each of these key-value pairs. The value, which is stored as a blob, can be any kind of data, like an image or document. Because the value is stored as a blob, it requires no upfront data modeling or schema definition. This also removes the need to index the data to improve performance. You just can't filter or control what's returned from a request based on the value, because the value is opaque.

Key-value stores use get, put and delete commands instead of a query language, which means the path to retrieve data is a direct request to the object in memory. The relationship between data isn't calculated, so there is no optimization overhead. You don't need to worry about

where to store indexes, network speed or balancing on a distributed system. Because of this simplicity, a key-value store is very fast and flexible, simple to use, highly scalable and portable. Redis is an open-source (BSD-licensed), in-memory, key-value data structure store that's used as a database, cache and message broker.

One of the advantages of using Redis comes from using the common Redis primitives, like `LPUSH`, `LTRIM` and `LREM`. Using those primitives allows tasks that are difficult and slow with traditional data stores to be accomplished much more easily. For instance, in a web app, deleted articles can be removed from the cache using `LREM`, or you can use `LPUSH` to insert a content ID at the head of the list stored at a key to show the latest item listings in a home page, and you can use `LTRIM` to limit that number of items in the list. With these simple primitives, Redis makes a developer's job much easier.

Because of its simplicity, speed and low latency, Redis is also a great solution for e-commerce app development when you are interested in efficiently storing user profiles and preferences, say for making product recommendations based on what users are viewing, or presenting real-time ads and coupons that are individualized to a customer's buying habits. Since all data is in memory, any delays in finding that data are eliminated, resulting in extremely high-speed performance. By using Redis as a cache in front of another DB, for example, huge gains are realized in speed.

Personally, I appreciate the online support available for Redis as well. It includes a complete list of commands as part of a detailed programming guide, along with multiple tutorials, administrative guides and other developer

resources. For a complete and detailed understanding of the massive benefits offered by Redis, visit https://redis.io.

GPU Acceleration with Kinetica Again, Kinetica is not an open-source database, but part of the OpenPOWER Foundation's open hardware/software ecosystem. Kinetica is a distributed, in-memory database that is accelerated by graphics processing units (GPUs). A GPU is simply a circuit designed to accelerate the creation of images for display by quickly altering and manipulating memory. Where a CPU has a many cores and a lot of cached memory, a GPU has thousands of cores, which results in speedups of more than 100x that of a CPU in some cases. Because they are great at taking large amounts of data and performing the same operation over and over, GPUs originally were intended for 3D game rendering. More recently, GPUs have been put to use speeding up computational workloads in functions like financial modeling, research, energy exploration and artificial intelligence. In fact, because of the parallel processing architecture that produces up to 100x faster processing speeds, Kinetica is ideal for analyzing big, streaming data. This makes it perfect for the predictive analysis that defines AI workloads.

Kinetica leverages GPU processing power to manage huge datasets, particularly streaming data, in a fraction of the time and on a much smaller hardware footprint than traditional databases. This is especially useful in IoT applications and geospatial visualization. It comes with visualization tools that are capable of rendering very large amounts of data, and there is no need to prepare the schema before the data can be analyzed. Kinetica is a great partner tool for transactional systems, data warehouses and

data lakes, and because it's also fully SQL-compliant, it's easy to query. It also supports REST, JSON, Java, JavaScript, C++ and Python, among others, making it very developer-friendly. This means you can explore huge datasets faster than ever without having to learn new query or programming languages or build new data models.

At this point, I've looked at several database offerings from the OSDBMS landscape. In the non-relational category are MongoDB, with its document-oriented data model; Neo4j, with its graph model; and Redis, offering a key-value approach. In the relational camp are EDB Postgres, an excellent analytics database, and an in-memory option, Kinetica. The common benefit of all these database management systems is speed, which is critical when talking about developing Big Data analytics applications.

Now let's turn, finally, to a platform that's ideal for hosting these OSDBMS and the apps built upon them: OpenPOWER LC servers that were designed from the ground up for Big Data by IBM and its OpenPOWER Foundation partners.

Why Implementing OSDBMS on IBM's OpenPOWER Systems Is the Right Approach

Leveraging the capabilities of OSDBMS to create truly innovative solutions requires not only processing speed, but also collaboration. Earlier in this ebook, I mentioned the OpenPOWER Foundation, a consortium of more than 250 members including some of the world's biggest names in technology—IBM, Google, NVIDIA, Mellanox Technologies, Tyan, Xilinx and Canonical. The OpenPOWER Foundation has

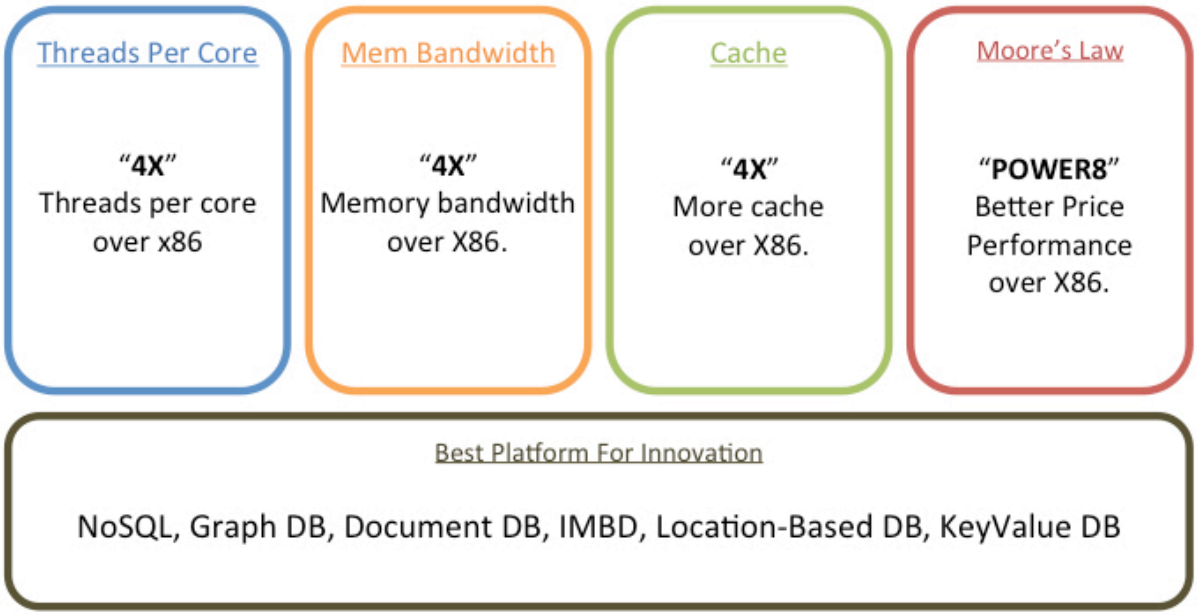

FIGURE 4. How IBM POWER8 Is Made for OSDBMS

been collaborating for years on system designs based on and around IBM's POWER processor architecture. You'll find one of the latest commercially available manifestations of that collaboration in IBM's OpenPOWER LC Servers.

IBM's OpenPOWER LC Servers with POWER8 processor technology were designed for Big Data workloads including the OSDBMS solutions I've been discussing here. IBM POWER8 delivers four times more processor cache, memory bandwidth and multithreading than commodity platforms can provide.

IBM POWER8 runs industry-standard Linux from Red Hat, SUSE and Canonical. This makes moving x86 Linux applications to Power more attractive and simpler than ever. Linux on Power provides the innovative platform developers truly require to leverage the power and scale of OSDBMS for data-intensive apps.

The POWER8 design combines computing power, memory bandwidth and I/O throughput to produce the speed required for Big Data and analytics workloads. POWER8 is designed to deliver four times more threads per core versus commodity infrastructure, four times more memory bandwidth and higher memory capacity versus commodity infrastructure, with scale-out systems that can deliver up to two terabytes on a two-socket server and all the way up to 16 terabytes for enterprise scale-up servers. POWER8 also provides 4X cache per processor at a lower latency, which allows you to process more data faster.

MongoDB For MongoDB, this is great, because you get a platform that delivers an integrated, real-time view of all your data. According to IBM, MongoDB on POWER8 provides 40% better performance per server than Intel Xeon. That's an excellent solution to data-center server sprawl, and if also taking into consideration deployment costs, MongoDB on POWER8 delivers twice the performance per dollar compared to x86-based systems—great news if you're looking to save the business money for future innovation.

EDB Postgres Advanced Server EDB Postgres Advanced Server also runs on little-endian Linux on POWER8, which removes portability issues. Running EDB Postgres Advanced Server on IBM's OpenPOWER LC Servers provides high-performance multi-threading, more cache, greater bandwidth for data, and about two times better price performance than on x86-based systems. IBM benchmarks have shown OpenPOWER LC Servers producing 60% better performance per core over Intel Xeon. Again, this is a perfect opportunity for your business to deploy

Through this solution, which works with any Redis client without changes to the standard Redis API, a single POWER8 server with CAPI-Flash acceleration can process more than 200K ops/sec with sub-millisecond latency—that's fast.

more workloads on less and spend less on infrastructure deployments for your Big Data apps, leaving more resources left over for innovation.

Redis Also a member of the OpenPOWER Foundation, Redis Labs and IBM Power Systems collaborate closely to deliver a Redis solution optimized for POWER8 and its Coherent Accelerator Processor Interface (CAPI) as part of Redis support for the IBM Data Engine for NoSQL, which runs Redis on the IBM 840 Flash System, the IBM CAPI-Flash card and Redis Labs Enterprise Cluster (RLEC) for Flash software as a RAM replacement. Through this solution, which works with any Redis client without changes to the standard Redis API, a single POWER8 server with CAPI-Flash acceleration can process more than 200K ops/sec with sub-millisecond latency—that's fast. It also can store 90% of a multi-terabyte dataset on Flash and only 10% on RAM. When compared to a pure RAM-based Redis solution, this helps to reduce deployment costs by more than 70%. Redis tests have shown IBM OpenPOWER LC Servers performing 67% better than x86.

Neo4j Neo4j on IBM's OpenPOWER LC servers provides the world's most scalable graph database platform, capable of storing and processing incredibly large graphs. One of the big challenges in graph processing at scale is how to handle dataset size without compromising real-time capabilities. With the available 56TB of extended memory on the LC server using CAPI and Flash as described above, the size of real-time queries is increased, because the size of graphs that can be stored in memory has been increased. But, the LC server line with POWER8 is balanced as well. Each core can handle eight hardware threads at the same time, for a total of 96 concurrent threads on a 12-core chip. On-chip memory controllers enable high bandwidth to memory and system I/O. With CAPI acceleration enabled on an IBM OpenPOWER LC Server, this delivers almost two times the performance over Intel Xeon.

Kinetica Getting the best performance from Kinetica really depends on how fast data moves between CPU and GPU, because Kinetica is designed to leverage system memory. New NVIDIA NVLink Technology and IBM POWER8 provide the most advanced and most affordable approach to providing high-performance analytics with Kinetica. Interconnects, such as NVIDIA NVLink, open a wider path between the CPU and GPU, which enables Kinetica to take full advantage of system memory. GPU processing is no longer limited to the rate at which data can be moved through the I/O subsystem, which opens Kinetica up to larger datasets.

Kinetica has been shown by IBM to produce a whopping 2.5 times the throughput running on an IBM OpenPOWER LC Server with NVLink than on a similar x86 system.

Conclusion

It's a big, complex world out there, generating big, complex data. Traditional databases, development approaches and processing platforms aren't going to provide the performance necessary to deliver on today's demands for real-time data and analytics capabilities.

OSDBMS offer innovative developers several benefits—performance and flexibility for today's diverse data deluge—and perhaps more important, the advantages of open development models, including access to a vibrant community that adapts to change and real-world problems quickly and effectively. Not only do we find responsive and accelerated capabilities in OSDBMS, but enterprise-class tools and freedom from the constraints of proprietary solutions make OSDBMS more usable and accessible.

As the demands placed on app developers continue to evolve, apps themselves must continue to innovate in response to those demands. The open ecosystem of OSDBMS provides the environment necessary for ideas and innovation to flow in real and usable solutions. Through close collaboration, IBM and leaders in the OSDBMS space are providing industry-leading platforms for making the most out of these new database technologies.

Find out for yourself how you can benefit from developing your application with OSDBMS on POWER by following this link: https://www-01.ibm.com/marketing/iwm/dre/signup?source=mrs-form-12148&S_PKG=ov53321.■