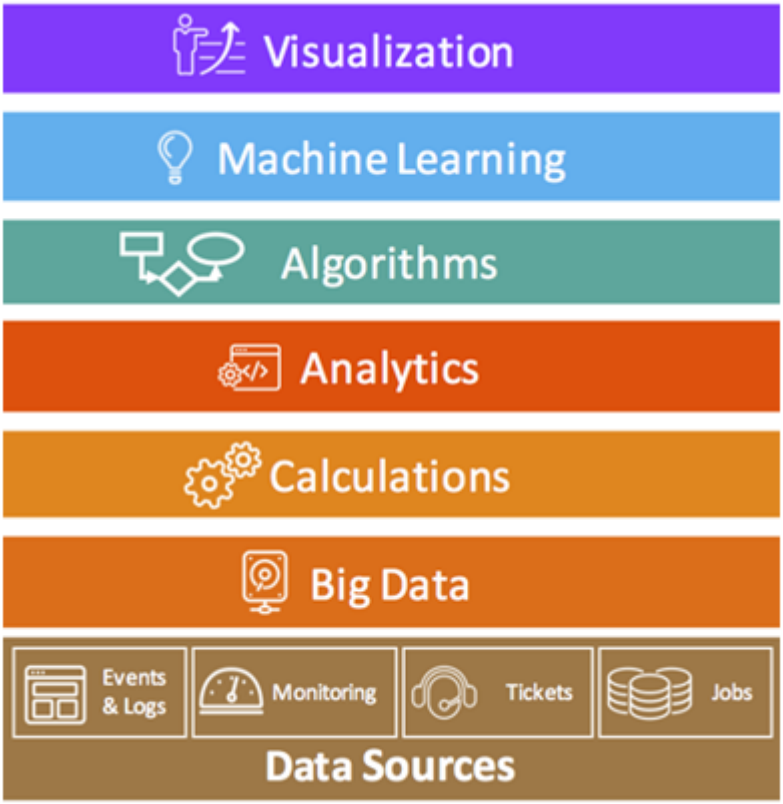
WHY AIOPS NEEDS BIG DATA AND WHAT THAT MEANS FOR YOU

In a previous blog post about AIOps, we presented the following graphic:



In this blog post, we'll go deeper into the Big

Data layer, explain why it is essential for AIOps, provide a brief overview of big data evolution and discuss what this means for your AIOps initiative.

Relational Databases and the Pre-Digital Age

Prior to about 20 years ago, most business, IT, scientific, health care and other systems didn't produce enough data to be what we now consider 'big data'.[i] What data they did produce could be stored in the standard database technology of the time – relational databases. Analytics could be built into tools with a relational database back-end or in standalone analytics platforms that leveraged relational databases.

In a relational database, data is organized into tables that consist of columns and rows. Tables typically define entity types (customer info, sales info, product info, etc.) and columns define attributes or properties. Rows contain data about specific instances of the entity. Entities are 'related' by virtue of being in the same table; tables are 'related' to each other using keys or indexes in order to associate data across different tables for the purpose of calculation, reporting and analysis. Queries that involve large numbers of tables and/or lookups across tables can severely stress the processing capability of the system. The challenge with relational databases is tuning performance to support the specific uses required.

Tables in relational databases must be defined in advance based on the data you are going to store in them. Understanding data type, size and expected use in retrieval, reporting, calculations and analytics is critical to relational database design. **You must understand the data structure, relationships between data entities and what you intend to do with it to get the expected benefit from querying in a relational database.** New data or changes to incoming data structure, relationships or uses necessitate changes to the database design.

Relational databases have a significant cost vs. performance tradeoff as data volumes and uses grow. While relational database reliability has generally been very good, performance is an ogoing challenge as tables proliferate, grow in size and support simultaneous query requests. Commodification of database technology in conjunction with drastic reductions in the cost of storage and processing helped organizations deal with scalability issues but structurally, the technology was never going to be able to support what we have come to call 'big data'.

Big Data

Beginning around the year 2000, we began to see an explosion in data creation thanks in part to a transition from proprietary, expensive, analog storage to commodified digital storage. Digital tape meant more data could be archived. CD/DVD (and then Blu-ray) meant more data could be distributed and shared. Increasing PC hard drive, processor and memory capacity meant more data was being captured and stored by individuals.

Once these technologies permeated institutions like hospitals, companies and universities, digital data generation and collection exploded. People started to ask how to unlock the business, scientific and knowledge value of these massive data sets. Two technical challenges needed to be overcome: the rigidity of relational data structures and scaling issues for processing queries on relational databases.

The first problem was addressed with the development of 'data lakes'. A data lake is a single store

for structured data such as that in relational databases (tables, rows, columns) but also semi-structured data (logs, JSON), unstructured data (emails, documents) and other data (images, audio, video).[ii] Data lakes collect all data, regardless of format and make it available for analysis. Instead of doing extract, transform, load (ETL) tasks or applying a data schema at ingestion, the schema is applied when the data is called upon for analysis.

The second issue was addressed with massively parallel processing (MPP). Relational databases rely on single or shared storage, which can be accessed by many processing nodes. The storage becomes a bottleneck due to either performance or query queuing. Simultaneous queries on the same data may have to be queued – to wait – on other queries to make sure they are using the most updated data.

MPP attempts to segment data across processing nodes, eliminating the single storage bottleneck. Segmentation is done by data type, expected use or even as sub-segments of larger data sets. This permits simultaneous or "parallel" processing that enables significantly increased query performance over traditional relational databases.

Of course, MPP segmentation presents its own challenges as well as the need for segmented data reconciliation. However, for relatively static data of the sort typical for early big data analysis, this approach worked well. Queries could be batched and executed in parallel vs. serially and doing complex analysis on massive data sets became achievable for most organizations.

The implementation of data lakes and MPP is best exemplified in Apache Hadoop – a technology with which most technologists are familiar – and specifically in Hadoop 1.0. Hadoop introduced the 'Hadoop Distributed File System' (HDFS) and MapReduce to address the limitations of traditional relational databases for big data analytics.

HDFS is an open-source data lake (accepting almost any data type as-is), supports data distribution across commodity hardware and is optimized for MPP queries that segment data. It made storing and utilizing massive data sets for analytics a technical and economic reality. MapReduce is an MPP engine designed to structure queries for parallel execution on segmented data in HDFS.

Apache Hadoop commoditized big data for organizations in every vertical. Scientists, business analysts, health science researchers and others began doing deep analysis on massive data sets to try and identify cures, weather patterns, insights, competitive advantage and more. The big data boom was born. But Hadoop 1.0 had some limitations that limited its utility for certain applications:

- MapReduce was the only application that could be used with HDFS
- MapReduce only supported batch processing of structured queries. You couldn't work with streaming (real-time) data or do interactive analysis.
- While relatively easy to setup and administer, optimization of data and queries was difficult. Organizations required data scientists to manage investigations to get useful results.

Cue Hadoop 2.0 - the democratization of big data and the enablement of AIOps.

Big Data for AIOps

With Hadoop 2.0, Apache released YARN ("Yet Another Resource Negotiator"). YARN sits alongside MapReduce and compliments its scheduling and batch capability with support for streaming data and interactive query support. YARN also opened the door for using HDFS with compatible, non-

Apache solutions.

Streaming, interactive big data analytics were now possible. Integrating 3rd party applications with Hadoop meant that vertical applications could incorporate a new level of analytics – if they were re-architected. Hadoop, however, was still difficult to optimize and use for organizations who needed an analytics practice but didn't have data science resources.

Enter market influence. Seeing the need for easier to use and more purpose-built solutions, companies like Elastic, Logstash and Kibana emerged ("ELK" or the 'Elastic Stack') offering batch, streaming and interactive big data analytics and ultimately becoming an alternative to Hadoop for some use cases

Why does this matter for core IT Operations and Service Management? Because both IT disciplines rely on streaming data and interactivity. In ITOM and ITSM applications, analytics had been limited by the database technology used and application architecture. And IT as a cost center couldn't justify hiring data scientists to find ways to use analytics for monitoring, remediation and service delivery use cases.

On the other side, the digital transformation of enterprises has been simultaneously revolutionizing the role of IT while applying unprecedented pressure on IT to deal with scale, complexity and speed. To support business innovation and keep up with digital transformation, IT needs systems that:

- Bring together diverse IT data
- Use machines to analyze massive amounts of streaming data in real-time
- Generate meaningful information for IT specific use cases (event management, alerting, workload placement, root cause analysis, cloud cost optimization, etc.)
- Identify additional automation opportunities
- Integrate with IT workflows and support interactive and historical analysis.

The challenge in transitioning to an analytics-based approach has been the limitations of purpose-built applications and their data silos. IT tools are not easy to replace or upgrade and even if they are re-architected to support big data analytics, their data remains siloed. Enter AIOps.

Fundamental premises of AIOps are:

- For IT to respond to digital transformation, machines must take over manual analysis
- Analytics must be real-time on streaming data as well as historical data
- Datasets must include diverse IT data from different silos
- Systems should be interactive, both from a technical and usability perspective

AIOps is only possible now with the commodification and evolution of big data technologies. It needs to support diverse data (data lakes); it needs to support simultaneous analytics on massive amounts of data; it must do analytics in real-time on streaming data; and must allow for interaction by human agents.

AIOps does not replace domain-specific ITOM and ITSM tools. It changes the approach to IT management by taking data from domain tools, housing it in a big data backend, applying analytics and machine learning, and democratically distributing insights across the organization.

With this background in mind, here are the key implications for your AIOps initiative:

- You must choose either to build a big data backend yourself on Hadoop, ELK or some other technology - or rely on a partner delivered solution. Partner solutions may be big data as a service or a big data backend in an AIOps platform. You should not build an AIOps initiative around a traditional relational database.
- If you build the platform yourself, recognize that you will take on the technical debt associated with ensuring performance of the solution; maintenance of an elastic (public or private cloud) infrastructure; as well as the creation of a robust data science practice. That practice must deal not only with the analytics theory, but also the implementation of that theory in your AIOps platform (e.g. development in Python or R)
- Ensure that your domain IT tools have APIs that support streaming data to the AIOps big data backend or that you can provide near real-time ETL of critical data (e.g. tickets, events, logs, etc.) to the platform. AIOps analytics, correlation and pattern matching algorithms need synchronized feeds of diverse data.
- Prepare organizationally for the shift. Not only will different, typically siloed, IT groups be required to share data and collaborate, they will also need to agree on review and response processes. Data will need to be visualized in common and what counts as 'normal' and 'abnormal' may need to be redefined in the context of the new joint approach.

AIOps is *the* path forward for enterprise IT but it will not come overnight nor without serious investment of thought, time and resources from I&O leaders. This article addresses just one aspect of a successful AIOps initiative. Look for future articles on other key elements such as analytics and algorithms.

To help you with your AIOps initiative, BMC offers the TrueSight AIOps platform which leverages machine learning, analytics, and big data technologies to reduce MTTR and drive the digital enterprise. TrueSight is designed for enterprise IT organizations in digitally transforming businesses. To learn more about TrueSight AIOps, click here.

[i] Financial systems did (and do) but most relied (and continue to rely) on a different type of technology.

[ii] Campbell, Chris. "Top Five Differences between DataWarehouses and Data Lakes". *Blue-Granite.com*. Retrieved May 19, 2017.