ISITIX

Hardware and system for Big Data

Overview - 2019

ISITIX

Hardware and System
for Big Data

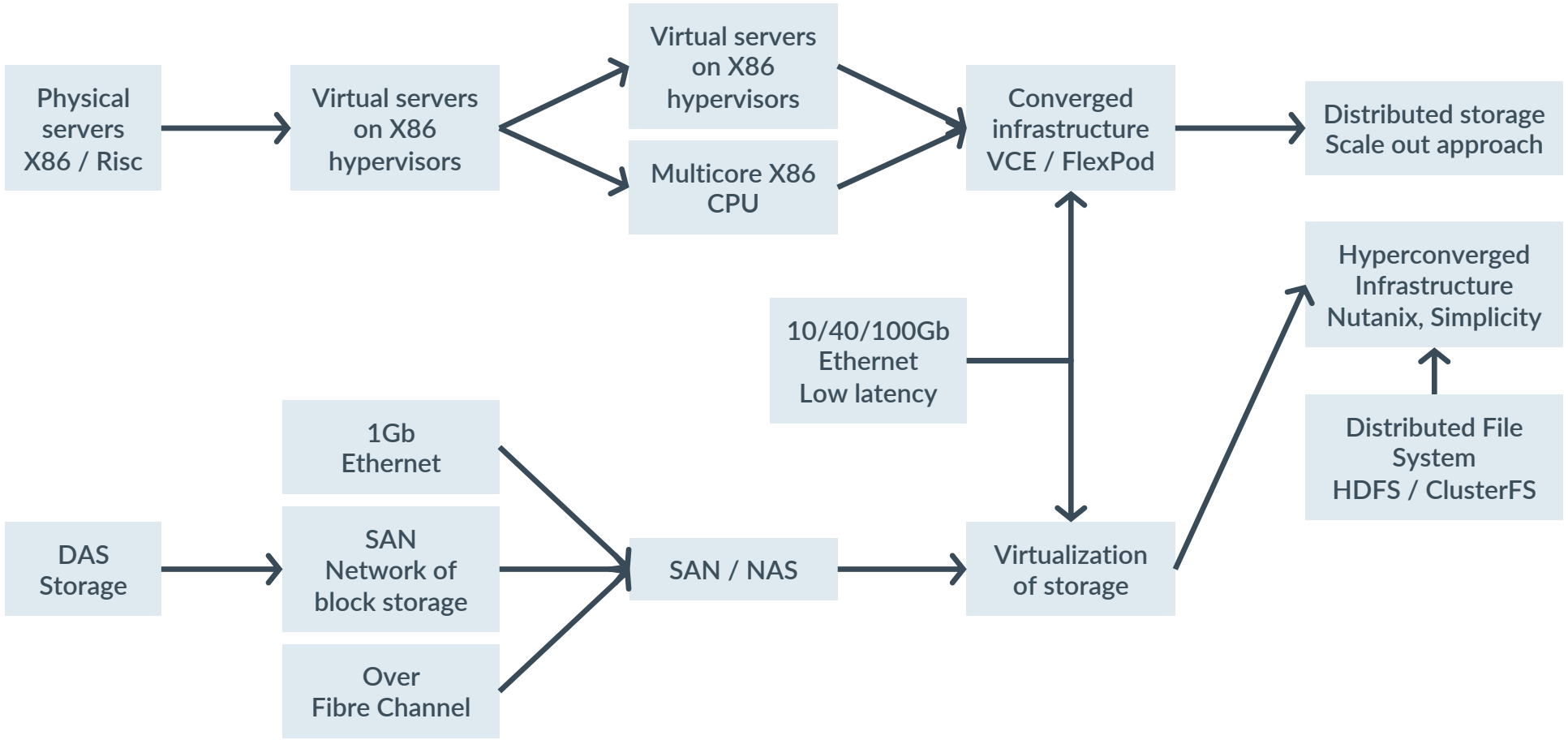
07/10/2019

2

Part 1

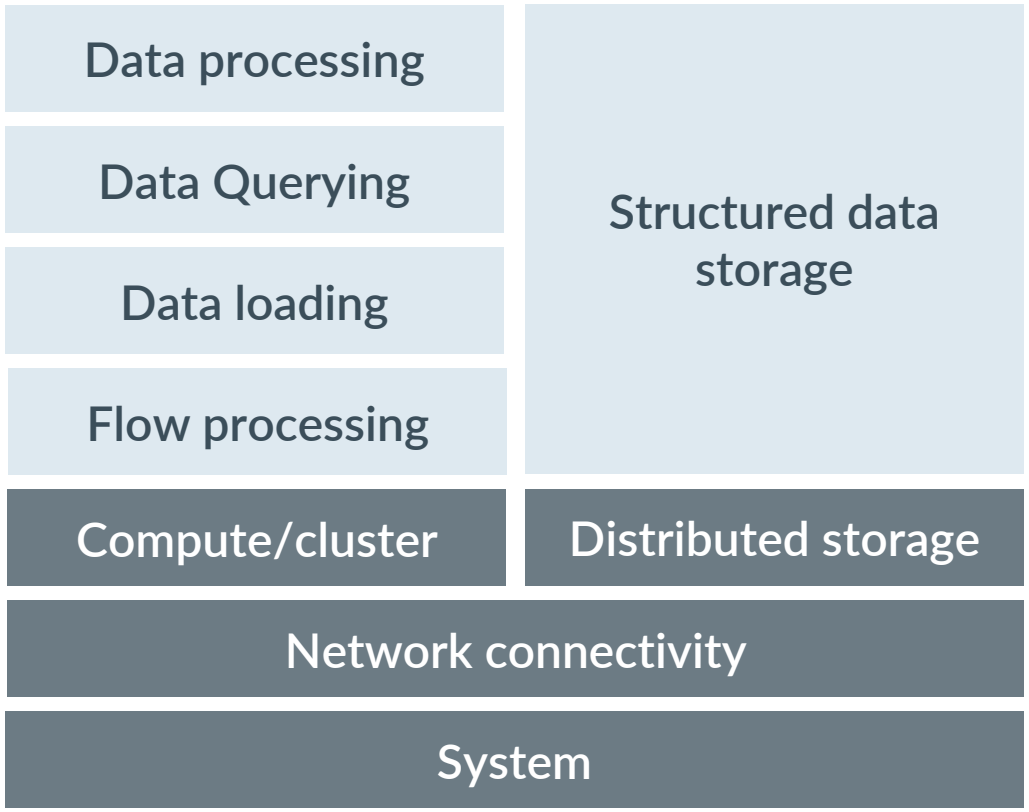
Infrastructure for Big Data

From X86 and DAS to hyperconverged and distributed infrastructure



Big data stack

Domain	Technologies
Clustering technologies	Mesos Kubernetes
Distributed storage	Hbase Cassandra Redis MongoDB CouchDB Node4J
Data querying interfaces	Phoenix Hive ...
Data processing	Spark Hadoop ...
Stream processing and data loading	Nifi Storm



Infrastructure blocks

Block	Option
Storage	Distributed Centralized Object / Software defined
Network	Converged Software Defined Network Ad hoc
Compute	Commodity Dedicated Specific / Heterogeneous Edge
System	Bare-metal Container VM Container on VM

Storage access time: what's new?

RAM access time : nanosecond

SSD disks access time : 0,1 millisecond

SAS disks access time : several milliseconds

Low latency switches with high throughput :

Microseconds and less for latency

10Gb/40Gb of throughput

Data locality is less crucial in modern datacenters but is still relevant for WAN distributed architecture

Low price of RAM, SSDs disks and performance of Ethernet switches ➔ architecture patterns for big data platform and database architecture in general have to be reconsidered

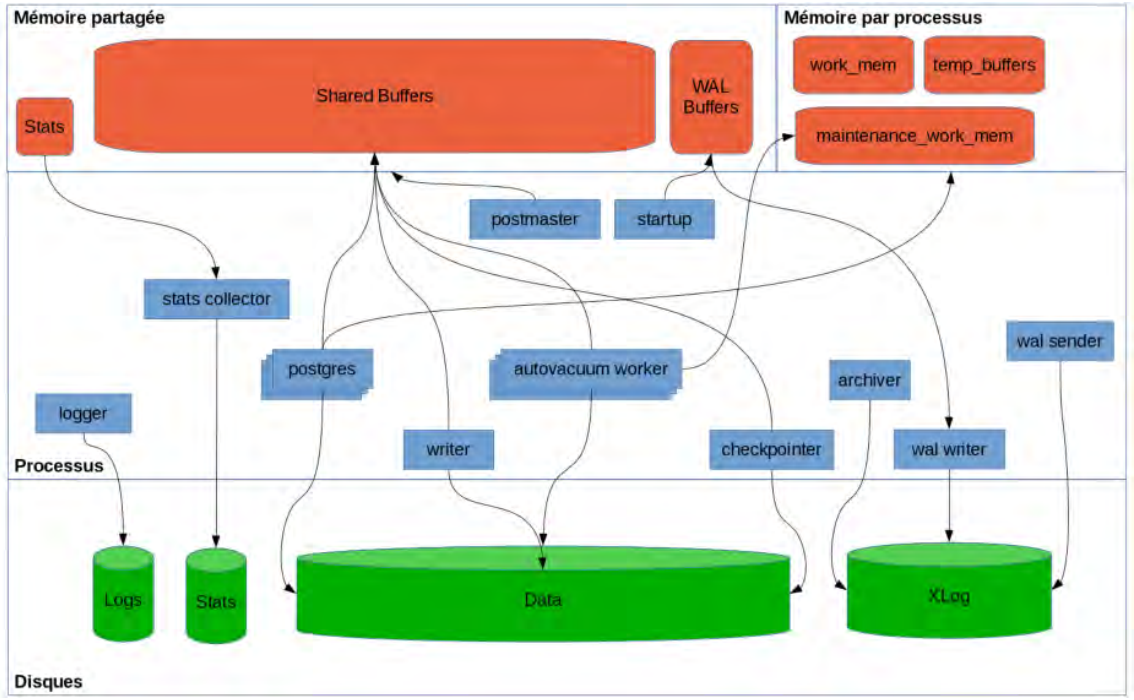
Key points of distributed architectures

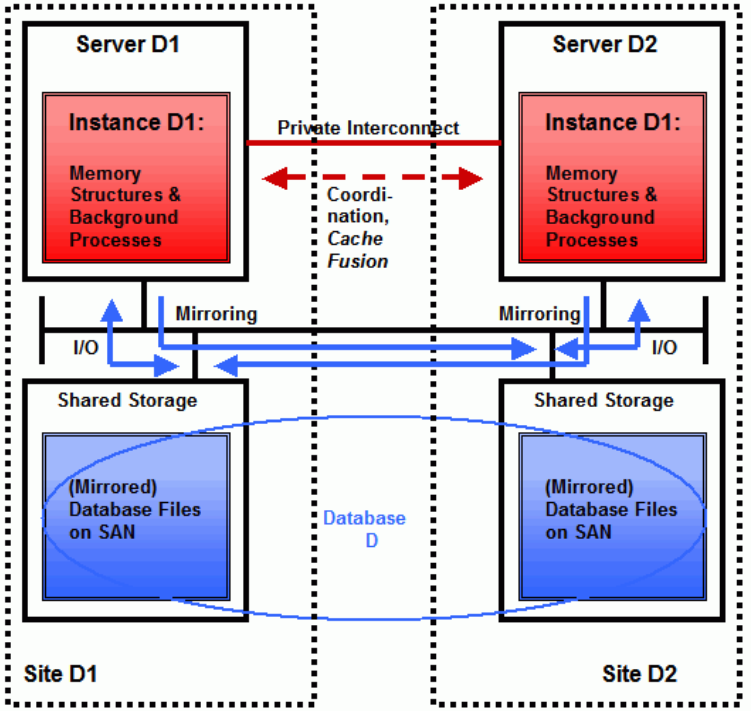
Domain	Key points
Data structure	Loosely defined structures Continuous evolution / application must deal with heterogeneous data schema
Data consistency	Immutability and versioning Distributed snapshot
Data distribution and partitioning	At least three copies Key management and data distribution depending on data access pattern
Query interface	SQL is back
Data storage	Solution dependent : DFS, SD storage, ...

Data architecture

RDBMS architecture

Postgresql : architecture et notions avancées
(Guillaume Lelarge)





ISITIX

RDBMS strength: data operation

Data integrity

Data consistency

Concurrency management

Cache management

Query optimizer

Backup and recovery tools, near RT RPO/RTO

Monitoring and diagnostic tools

SQL is a de-facto portable standard

But advanced features, materialized views, trigger, stored procedure, table partitions are not so portable

Broad adoption of SQL

Many platforms from standalone application (sqlite, access…) to servers (mysql ingres, db2, sqlserver, postgresql, oracle, …)

RDBMS weakness: development environment

No debugging tools

Data model and data are intricate

No easy way to experiment

No sandbox, No simple way to make a snapshot, run it apart and diff

Each developer have to work on its own DB instance

No versioning of data model, no diff, no patches

Going into production is painful...

Syncing code and data model is hard work

Design is not agile

Conceptual data model => logical data model => physical data model

RDBMS design patterns are not widely known

Apart from Backus normal forms and Kimball star schemas / snow flakes

When things go wrong and you have to dig into the system, it's often a ping-pong game between DBA and developpers

Programming paradigms / Programming languages

Characteristics	Key points
Performant	Huge datasets fit in memory
Dynamically typed	Why should you be more constrained by your database than by your programing language ?
Rich set of data structures	Key/value pairs Dictionnary Trees ...
Iterators and lambda	Conciseness Expressiveness
Rich libraries	Machine learning, text processing, visualization, ...

Data analysis directly from text files or loosely defined data structures (without loading data into a SQL database) becomes easier with this new languages and libraries

Design principles

1. **Scale-out approach rather than « legacy » scale-up approach: facilitate scalability/availability with common x86 servers**

2. **Bring the compute resources as close as possible to the storage resources based on the debatable assumption than network is often a congestion point during I/Os access**

3. **Distribute the load of compute and I/Os among several nodes**

Execution environment

1. **Most of the Hadoop's components services are executed using Java environnement**

2. **X86 servers Linux and Windows : take benefits of the current huge performance of modern x86 CPUs**

3. **Physical or virtual server**

4. **Trend: containerization (Docker) and clustering (Kubernetes)**

Storage

1. **Relies on a distributed file system:**
   HDFS (Hadoop Distributed File System)

2. **HDFS file system does not meet all POSIX constraints:**
   It was a condition to maintain high throughtut during acccess operations

3. **HDFS file system has been designed to handle large files (> GB):**
   there is huge overhead with small files on metada management level but workaround exist (compactation of files, hbase,…)

Storage

1. **Historically storage architecture was designed to handle batch loads. Local mechanical disks attached to each Data Node did the job: low latency during I/Os access was not a big priority**

2. **Today each Data Node can use local heteregenous storage resources: memory, SSDs disks or mechanical disks**

3. **Tiering approach is emerging on Hadoop platform to meet the different I/Os load profiles and needs: sequential requests, intercative requests , temperature of data (hot, warm and cold)**

Network

1. **Network is a critical point of a Big Data architecture: network is used to access data AND to manage the integrity/availability of the data blocks**

2. **Network outage can have a serious impact on data integrity level and can cause persitent loose of data:**
   big difference with « legacy » architecture

Hierarchical network is superseded by SDN flat network with tag isolation

Network

1. **Huge throughtput and low latencies of the new datacenter switches facilitate the emergence of big data architecture**

2. **Nevertheless oversubscription ratio between access layer and agregation layer have to be defined carefully**

3. **Current trend of deploying spine leaf architecture for new LAN datacenters can be a good answer for very big Hadoop cluster: less hops during data transport and synchronisation tasks**

Data Integrity

High level

1. Big Data Architecture is not currently designed for business applications which does not suffer any loose of piece of data
2. Several SPOFs exists during write operations: Name Node and Data Node (with default configuration)

Low level

1. Immutability
2. Versioning
3. Election
4. Versions pruning
5. Application layer

High availability

1. **Replication of block storage on HDFS level is achieved through the network among Data Nodes: default replication factor is 3**

2. **By default write acknowledgement is achieved when data block is written to only one data node**

3. **Secondary Name Node, Standy-by Name Node, Checkpoint Nodes, Backup Node or Snapshots are available features to reach a high level of availability**

4. **Rack awareness functionnality could be also a way to increase availability Low level**

5. **Stretched cluster**

Business Continuity and DRP

1. **Currently there is no advanced replication functionnality between HDFS cluster for DRP purpose. DistCp tool provides intra and inter clusters but does require manual configuration and can be painful to manage with big infrastructure**

2. **EMC² ECS could be an answer**

3. **Hbase provides advanced replication feature for disaster recovery between different geographical sites**

Deployment options

For preproduction environments, you may prefer to use virtual servers or containers running on classical infrastructure to minimize investments at short term

Which modes for node deployment of the Big Data cluster ?

1. Full virtual servers based on « legacy network storage architecture »: easy to manage and to deploy but what about the constraints on physical layer and specifically on storage layer. A high maturity on QoS hypervisor layer is required, for VMs placement rules and for storage resources allocation
2. Hybrid mode: dedicated physical server with DAS storage with an intermediate virtualization layer of OS to facilate deployment and management
3. Full physical mode: can be painful to deploy and to maintain with the physical constraint on OS layer. TCO should not be minimized in that case and automatic deployment through scripts is mandatory
4. Cloud mode: deploying on off-the-shelf cloud service

Several deployment principles

Dockers container is an additional virtualization layer which can help to reduce the number of servers whatever the deployment mode: virtual, hybrid or physical

As already mentionned, the network layer is a key point of a big data architecture infrastructure. For big deployment a dedicated infrastructure is required with perfomance switches (cf. high bandwidth and low latency)

Monitoring

1. Hadoop provides natively several metrics and interfaces to monitor a big data platform : Ambari / YARN / Hbase…
2. Nagios plugs-in exist or can be developped to monitor Hadoop clusters
3. Other several tools : Zabbix, Ganglia, Splunk, Sequence IQ…
4. Don’t forget to monitor every physical infrastructure components with traditional monitoring tools : servers, network and storage

Upgrade tasks

1. HDFS provides tools to securely operate upgrade tasks
2. With HA configuration upgrade can be achieved without downtime

Backup operations

1. **A big data platform with more than 100 To of data is very difficult to backup during a classical maintenance window (~24h)**

2. **Usage of native checkpoints and snapshots on HDFS level is a way to guarantee a first level of integrity**

3. **DistCp tool may also provide an additional level of protection**

4. **Critical data sets can be still extracted and secured through classical backup software**

Final thoughts

Non deterministic

State is distributed

No centralized truth

Unstoppable

Part 2

Review of articles about computer architecture

A New Golden Age for Computer Architecture

Article written by JL.Hennesy et DA.Patterson in ACM communication (2019)

https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext

Moore's law is over: sequential execution performance doesn't improve anymore

Dennard scaling is over: we face the heat barrier

Processor level optimization (speculative execution) wastes many CPU cycles (10% to 40%)

Processor level optimization is a target for security breach (meltdown, spectre…)

Libraries (such as python libraries) are not optimized for performance (To do)

Domain-specific hardware is playing a major: TPU, GPU, FPGA

The era is full of opportunities to renew, redesign computer architecture

C is not a low level language

ACM communication July 2018, D.Chisnall

C is not a language for parallelism and doesn't fit well with recent processors that implement parallelism natively

We keep on working with an abstract model of processor that was designed for PDP11 system a very long time ago

There is room for new natively parallel system languages

Evaluating and Optimizing I/O Virtualization in Kernel-based Virtual Machine (KVM)

Timer interrupt may significantly impact I/O performance in virtualized environments

Article published by https://link.springer.com/content/pdf/10.1007%2F978-3-642-15672-4_20.pdf

ISITIX

Hardware and System for Big Data

07/10/2019

A Domain-Specific Architecture for Deep Neural Networks

ACM communication, September 2018

Article written by Google engineers to compare TPU and GPU

Roofline model

TPU beats both K80 Nvidia GPU and Haswell Intel processor on every benchmark

But TPU is on available as a service on Google Cloud…

Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network

Sigcomm paper

Article written by Google engineers about Clos Topologies:
http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p183.pdf

Requirements

Bandwith
Cost
Scaling
Robustness

Design principles

Clos topologies
Centralized control
No hierarchy
No boundaries

The tail at scale

ACM Communication February 2013

Article written by Google engineers about how to manage response time variability

“ Even for services with only one in 10,000 requests experiencing more than one-second latencies at the single-server level, a service with 2,000 such servers will see almost one in five user requests taking more than one second (marked “o” in the figure).”

A small group of outliers may impair the quality of service for a much larger group of users

Time is an illusion
Lunchtime doubly so.

ACMQueue January 2016

Leslie Lamport's "Time, Clocks, and the Ordering of Events in a Distributed System" (1978), and only a few more have come to appreciate the problems they face once they move into the world of distributed systems.

The relative nature of time

Synchronization vs syntonization

Synchronization = exactly the same moment (whatever the way to measure it)

Syntonization = exactly the same time tick (~frequency)

Mikael Dautrey
+33 6 61 44 32 66 — mikael.dautrey@isitix.com

Alan Guillais — alan.guillais@isitix.com

www.isitix.com