Designing K-Means for the cloud

Matthieu Durut, Fabrice Rossi

March 6, 2012

Few reminders about distributed K-Means

$$\begin{aligned} KMeans\ Sequential\ Time &= (3Knd + Kn + Kd + nd) * I * T^{flop} \\ &\simeq 3Knd * I * T^{flop} \end{aligned}$$

SMP Cost

$$
\begin{aligned}
KMeans\ SMP\ Distributed\ Cost &= T_P^{comp} \\
&= \frac{(3Knd + Kn + Kd + nd) * I * T^{flop}}{P} \\
&\simeq \frac{3Knd * I * T^{flop}}{P}
\end{aligned}
$$

DMM Cost

$$
\begin{aligned}
&KMeans\ DMM\ Cost \\
&= T_P^{comp} + T_P^{comm} \\
&= \frac{(3Knd + Kn + Kd + nd) * I * T^{flop}}{P} + T_P^{comm} \\
&\simeq \frac{3Knd * I * T^{flop}}{P} + O(log(P))
\end{aligned}
$$

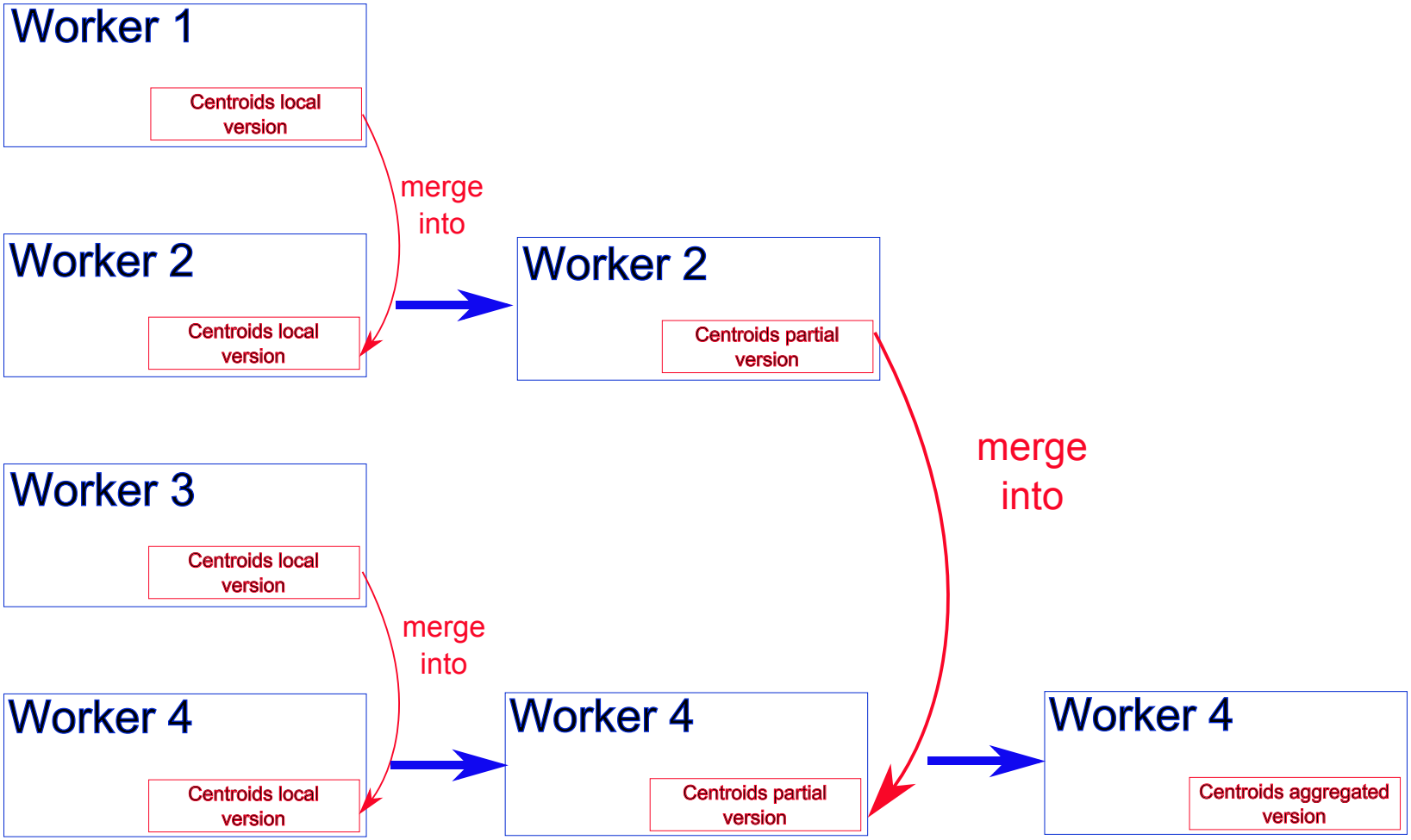

Figure: Merging centroïds logic

classical issue : pipes

The size of the biggest pipe (on top of the tree) and the aggregated bandwidth.

modelisation of T_P^{comm} in DMM

$$T_P^{comm} = \sum_{p=1}^{\lceil log_2(P) \rceil} (2 \frac{lKd * SoD}{B_{\frac{P}{2^p}}^{DMM,MPI}} + 5lKd * T^{flop})$$

modelisation of T_P^{comm} in DMM (2)

$$T_P^{comm} = (2\frac{lKd * SoD}{B_{P,avg}{}^{DMM,MPI}} + 5lKd * T^{flop}) * \lceil log_2(P) \rceil$$

SpeedUp

$$
\begin{aligned}
Speedup &= \frac{KMeans\ Sequential\ Cost}{KMeansDMMDistributedCost} \\
&= \frac{KMeansSequentialCost}{T_P^{comp} + T_P^{comm}} \\
&= \frac{3nKdIT^{flop}}{\frac{3nKdIT^{flop}}{P} + \left(2\frac{IKd*SoD}{B_P^{DMM,MPI}} + 5IKdT^{flop}\right)\lceil log_2(P)\rceil} \\
&= \frac{3nT^{flop}}{\frac{3nT^{flop}}{P} + \left(2\frac{SoD}{B_P^{DMM,MPI}} + 5T^{flop}\right)\lceil log_2(P)\rceil}
\end{aligned}
$$

Some comments

- RAM is not modelize $\Rightarrow$ speed up might be greater
- speedup independant of K, I and d
- winner paper model !

$$\lim_{n \to +\infty} SpeedUp = P \tag{1}$$

$$\tag{2}$$

First case study : EDF

- n = 20 000 000 time series
- d = 24 hours * 365 days * 10 years = 87600 values per series
- K = $\sqrt{n}$ = 4472 clusters
- Note : volume of timeseries is 12.3 TeraBytes.

1 monocore

37 years

Yahoo terasort winner

- 910 nodes
- 4 dual core Xeons @ 2.0ghz per a node
- 1 gigabit ethernet on each node
- 40 nodes per a rack
- 8 gigabit ethernet uplinks from each rack to the core

Yahoo terasort winner(2)

$$T_P^{comp} = \frac{(3nKd + nK + dK + nd) * I * T^{flop}}{P}$$
$$= 322893 seconds$$

$$T_P^{comm} = (2 * \ Broadcast\ time\ between\ 2\ processors + Merging\ time) * \lceil$$
$$= (2\frac{IKd * SoD}{B_{P,avg}^{DMM,MPI}} + 5IKd * T^{flop}) * \lceil log_2(P) \rceil$$
$$= 225866 seconds$$

Yahoo terasort winner(2) bounded by aggregated bandwidth

$$B_{P,avg}^{DMM,MPI} = 109 Mbits$$

$$\begin{aligned} T_P^{comm} &= (2 * \ Broadcast\ time\ between\ 2\ processors + Merging\ time) * \lceil \\ &= (2\frac{IKd * SoD}{B_{P,avg}^{DMM,MPI}} + 5IKd * T^{flop}) * \lceil log_2(P) \rceil \\ &= 434155 seconds \end{aligned}$$

Pricing on Azure

- Storage price : 1845 dollars/Month (0,15 dollars /GB/Month)
- No charge for internal transfers ?
- update : 10 GBytes/Month : 3 dollars (0.3dollars /GB)
- cost of running the KMeans : (very rough) 0.12dollars/hour * 324120 = 40 000dollars
- storage requests : ? from 0.01 dollars to 11 millions dollars

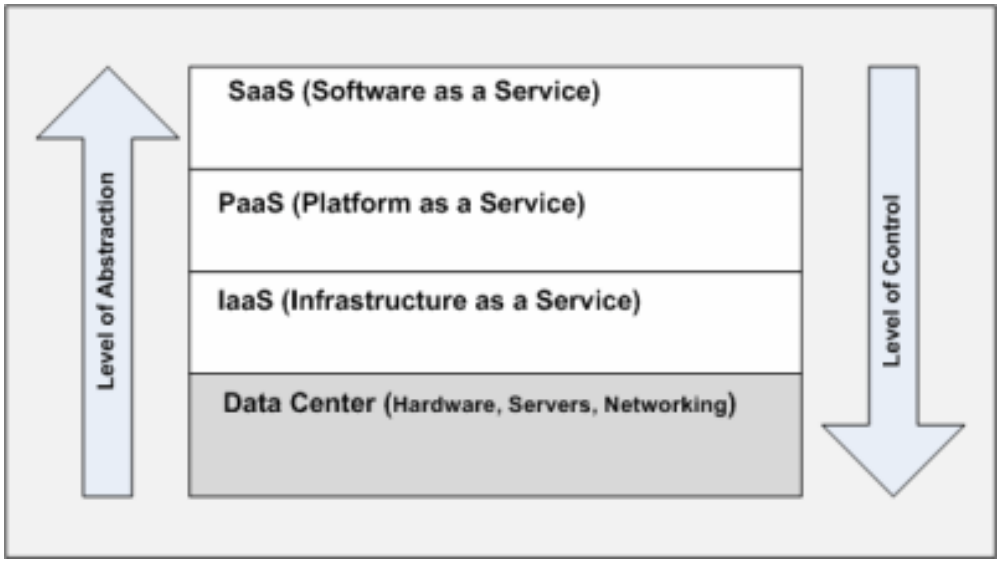

Figure: Paas, Iaas, SaaS

Paas and Iaas : some tradeoffs

- abstraction/control tradeoff
- scalability/cost to develop tradeoff

Introducing Lokad.Cloud

- higher level abstractions

Azure Abstractions

- Azure Queues
- Azure BlobStorage
- Azure TableStorage
- Azure SQLStorage

Azure Queues

- loosely coupled asynchronous communication mechanism
- not scalable yet (in size and number of messages)
- FIFO when not sought
- messages can be processed several times if worker fails (see idempotency)

Azure BlobStorage

- objects up to 50 GB
- consistency ?
- Key/value system

Azure TableStorage

- Tables/Entities/Properties
- key/value pair also

Azure QueueServices

- SOA
- Queue consumers

ACID/BASE constraints on Cloud

- ACID
- BASE
- CAP Theorem
- CLAP system

ACID/BASE constraints on Azure

- Availability : No
- Internet and Electricity neither
- Consistency : ?

Consistency on Azure : replicas and consistency

- Block commits
- Etag
- very few overwrites

Consistency on Azure : transactionality

- The SQL trick

Azure tradeoffs choices

- read/write tradeoff
- Responsivity/efficiency
- Synchronous replication/Asynchronous replication ?

BlobStorage or TableStorage ?

- Partition Key and locality
- size of objects
- efficiency of Blobstorage for objects larger than 4Kbytes
- Price

Abstractions performances

- BlobStorage Read : 13MBytes/sec/worker. Aggregated bounded to 393 MBytes/sec
- BlobStorage Write : 5MBytes/sec/worker. Aggregated bounded to 124 MBytes/sec
- Direct worker communication : 55MBytes/sec on very large messages
- VM : from 1 to 8 cores
- Queues throughput : 500 messages/sec

Redesign of the algorithm

- SOA
- No Master
- All workers can be switched
- No worker/data per affinity but ...
- Scaling up as an initiative in PaaS
- No exact workers count
- Pinging queues as a tradeoff between cost and simplicity
- Queue Messages count
- idempotent jobs

Redesign of the algorithm (2)

- Idempotent Counters
- Workers communication ?

Proposed Algorithm

- Scheme

Difficulty to anticipate speedUp

- Serialization/Deserialization. DataContract ? BinaryFormat ? GoogleProtoBuf ?
- variability of the bandwidth
- latency due to pinging queues
- a worker can fail, a VM can be killed
- Storage unavailability
- Stragglers issues ?

Is time to load data important ?

$$T_P^{Load} = \frac{nd * SoD}{P * \max(Bandwidth^{read}, \frac{MaxReadThroughput}{P})}$$

$$T_P^{Load} << T_P^{Comp}$$

Implicit supposition : loading is run once.

$$\frac{SoD}{3IK * T^{flop} * \max(Bandwidth^{read}, \frac{MaxReadThroughput}{P})} << 1$$

Communication modelisation

$$
\begin{aligned}
T_P^{comm,periteration} = & \textit{ Time to read 1 blob "simultaneously" by P workers} \\
& + \textit{ Time to write P blobs "simultaneously"} \\
& + \textit{ Time to read P blobs by 1 worker} \\
& + \textit{ Time to merge the different centroids version} \\
& + \textit{ Time to write 1 blob per 1 worker}
\end{aligned}
$$

Communication modelisation(2)

$$
\begin{aligned}
T_P^{comm,periteration} = & \frac{Kd * SoD}{Min(Bandwidth^{read}, \frac{BlobReadThroughput}{P})} \\
& + \frac{Kd * SoD}{Min(Bandwidth^{write}, \frac{MaxWriteThroughput}{P})} \\
& + \frac{PKd * SoD}{Bandwidth^{read}} \\
& + (P-1)Kd * 5floatOperations * T^{flop} \\
& + \frac{Kd * SoD}{Bandwidth^{write}}
\end{aligned}
$$

Communication modelisation(3)

$$T_P^{comm} \simeq \frac{IPKd * SoD}{Bandwidth^{read}}$$

SpeedUp

$$
\begin{aligned}
SpeedUp &= \frac{T_1^{comp}}{T_P^{Comp} + T_P comm} \\
&\simeq \frac{3IKdnT^{flop}}{\frac{3IKdnT^{flop}}{P} + \frac{IPKd * SoD}{Bandwidth^{read}}} \\
&\simeq \frac{3nT^{flop}}{\frac{3nT^{flop}}{P} + \frac{P * SoD}{Bandwidth^{read}}}
\end{aligned}
$$

Optimal Number of workers

$$P^* = \sqrt{\frac{3nT^{flop}Bandwidth^{read}}{SoD}}$$

Numerical Example

- $n = 432\ 000$
- $K = 657$
- $d = 168$
- $l = 100$
- $PMax = 64$
- $Bandwidth^{read} = 20 Mbytes/sec (resp. 5 Mbytes/sec)$

Results

$$Sequential\ KMeans\ time = 14300 seconds$$

$$Optimal\ number\ of\ workers = 57 (resp 28)$$

$$SpeedUp = 28,5 (resp 14)$$

Improvements : more reducers ?

scheme

Direct worker communication ?