Aplicacions dades massives

2024/2025

Desenvolupament d'Aplicacions de dades masives

- Grau Enginyeria de Dades
- curs 2024/2025

- Toni Espinosa, Ramon Grau, Joan Piedrafita, Christian Guzmán
- *Departament Arquitectura Computadors i Sistemes Operatius UAB*
- antoniomiguel.espinosa@uab.cat

Desenvolupament d'aplicacions de dades massives

- Toni Espinosa, Ramon Grau, Joan Piedrafita, Christian Guzmán

  antoniomiguel.espinosa@uab.cat

  ramon.grau@uab.cat

  joanjosep.piedrafita@uab.cat

  christian.guzman@uab.cat

- **Sesiones:** lunes 17 a 19, jueves 15 a 17.
- Contenidos de clase, laboratorios, datasets, materiales para revisar y entregas al Campus Virtual
- Grupo Teams asociado a clase para consultas

Sesiones teoria y lab

- Teoria
  - jueves 15 a 17 – Aula Q4/1009
- Laboratorios:
  - lunes 17 a 19 – Aula Q4/1009

Challenge-based mini-project

- Trabajo en grupo: 5 estudiantes
- Elección de un reto
- Entidades externas o proyecto propio: 3CAT, CAR Sant Cugat, Ajuntament Rubí, Accenture, ...
- Tres fases en dos asignaturas del segundo semestre
  - DAMD: Modelo de datos, Toni Espinosa
  - Computación En Nube: Servicios cloud y costes, Dani Franco
  - Visualización de Datos: Visualización de KPIs, Enric Martí (opcional)
- Por qué lo hacemos?

Desenvolupament Aplicacions Dades Massives - DADM 2025 - Aula Q4/1009				
Grup	Data	Hora		Prof
1	lunes, 10-feb.	17h-19h	Introducción a la asignatura, guía docente, planificación	TE
	jueves, 13-feb.		ACTIVITATS CAMPUS	
1	lunes, 17-feb.	17h-19h	Data models and query languages	TE
1	jueves, 20-feb.	15h-17h	Storage and retrieval	TE
1	lunes, 24-feb.	17h-19h	Data warehousing	TE
1	jueves, 27-feb.	15h-17h	Data warehousing	TE
	lunes, 03-mar.	17h-19h	Jornada MWC	
1	jueves, 06-mar.	15h-17h	Metodologia y liberación del reto	RG
1	lunes, 10-mar.	17h-19h	Case Study 1 - Definición problema	RG
1	jueves, 13-mar.	15h-17h	Case Study 2 - Estudio roles usuarios	RG
1	lunes, 17-mar.	17h-19h	Case Study 3 - Estudio roles usuarios	RG
1	jueves, 20-mar.	15h-17h	Presentación Cap Gemini/Case Study 4 - Propuesta	RG
1	lunes, 24-mar.	17h-19h	Case Study 5 - Propuesta	RG
1	jueves, 27-mar.	15h-17h	Presentaciones	RG
1	lunes, 31-mar.	17h-19h	Presentaciones	RG
1	jueves, 03-abr.	15h-17h	In-memory databases	TE
1	lunes, 07-abr.	17h-19h	Redis tutorial	TE
PLAB 1, 2, 3	jueves, 10-abr.	15h-17h	Redis LAB	JP/CG
	jueves, 24-abr.		Memenginy	
1	lunes, 28-abr.	17h-19h	examen parcial	
PLAB 1, 2, 3	lunes, 05-may.	17h-19h	Spark Dataframes	TE
PLAB 1, 2, 3	jueves, 08-may.	15h-17h	Spark LAB 1	JP/CG
PLAB 1, 2, 3	lunes, 12-may.	17h-19h	Spark LAB 2	JP/CG
1	jueves, 15-may.	15h-17h	Spark MLIB	TE
PLAB 1, 2, 3	lunes, 19-may.	17h-19h	Spark LAB 3	JP/CG
PLAB 1, 2, 3	jueves, 22-may.	15h-17h	Spark MLIB LAB 1	JP/CG
PLAB 1, 2, 3	lunes, 26-may.	17h-19h	Spark MLIB LAB 2	JP/CG
PLAB 1, 2, 3	jueves, 29-may.	15h-17h	Spark MLIB LAB 3	JP/CG
exm	viernes, 20-jun.	15:30	examen parcial	
exmrec	jueves, 26-jun.	15:30	reevaluación	

Contenido

- Introducción a las aplicaciones masivas de datos
- Conceptos fundamentales del tratamiento de los datos
- Sistemas para gestionar grandes volumenes de datos
  - In-memory databases: Redis
- Aplicaciones analisis de datos con grandes volumenes : Apache Spark
  1. Spark Dataframes
  2. Spark MLib
  3. Proyecte gestión de datos con Spark en Cloud

Planificación

1. Introducción
2. Modelos de datos
3. Almacenamiento y recuperación de los datos
4. Key-value storage systems
5. Estudio de casos de uso y modelos de datos
6. Bases de datos en memoria: Redis
7. Data analytics y Spark
8. Machine learning con Spark MLib

Evaluación

- 2 parciales de teoría + revaluación final
  - primer parcial: teoría hasta Redis (no incluido)
  - segundo parcial: Redis y Spark
- 3 entregas de prácticas (Redis, Spark, Spark Mlib)
- 1 presentación en grupo

- Nota final: Teoría * 60% + Entregas * 30% + Presentación * 10%

Teoría y entregas >= 4,5

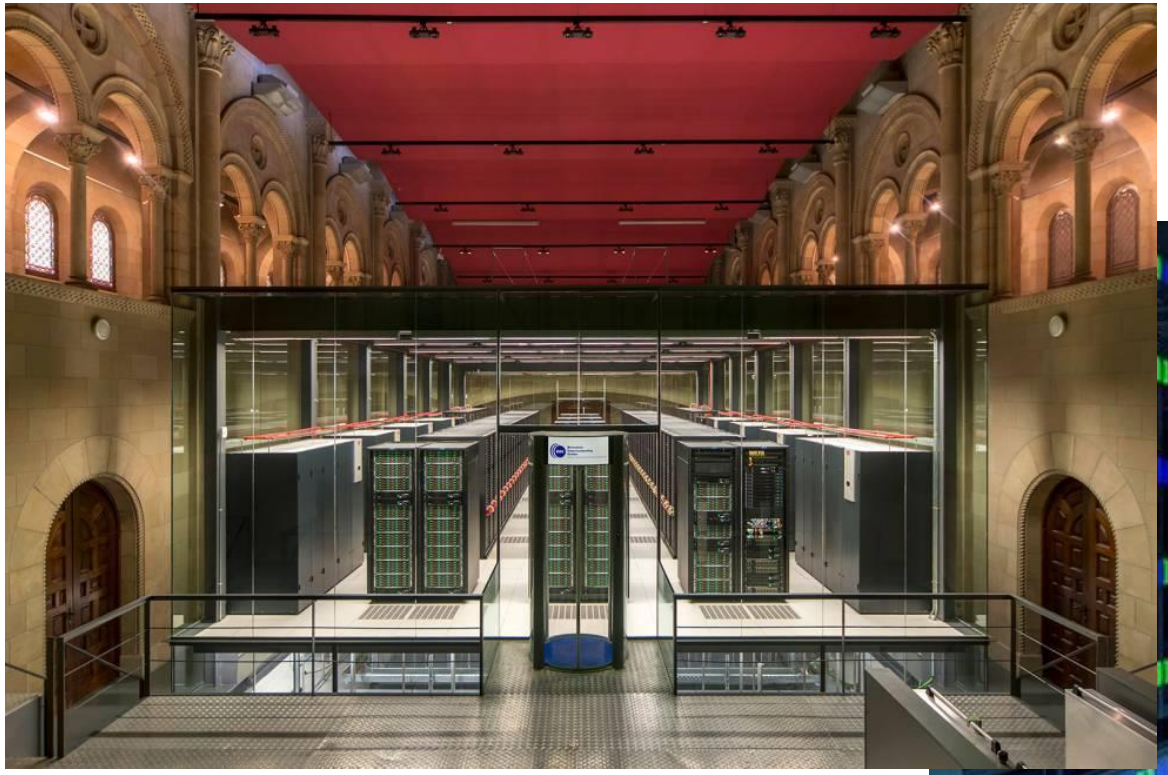
Objectives

- How can we describe **distributed systems**?
- Basic principles of distributed systems
- Massive data analysis paradigms

- How can we describe large data **applications**?
- Execution models: batch, streaming, en memoria

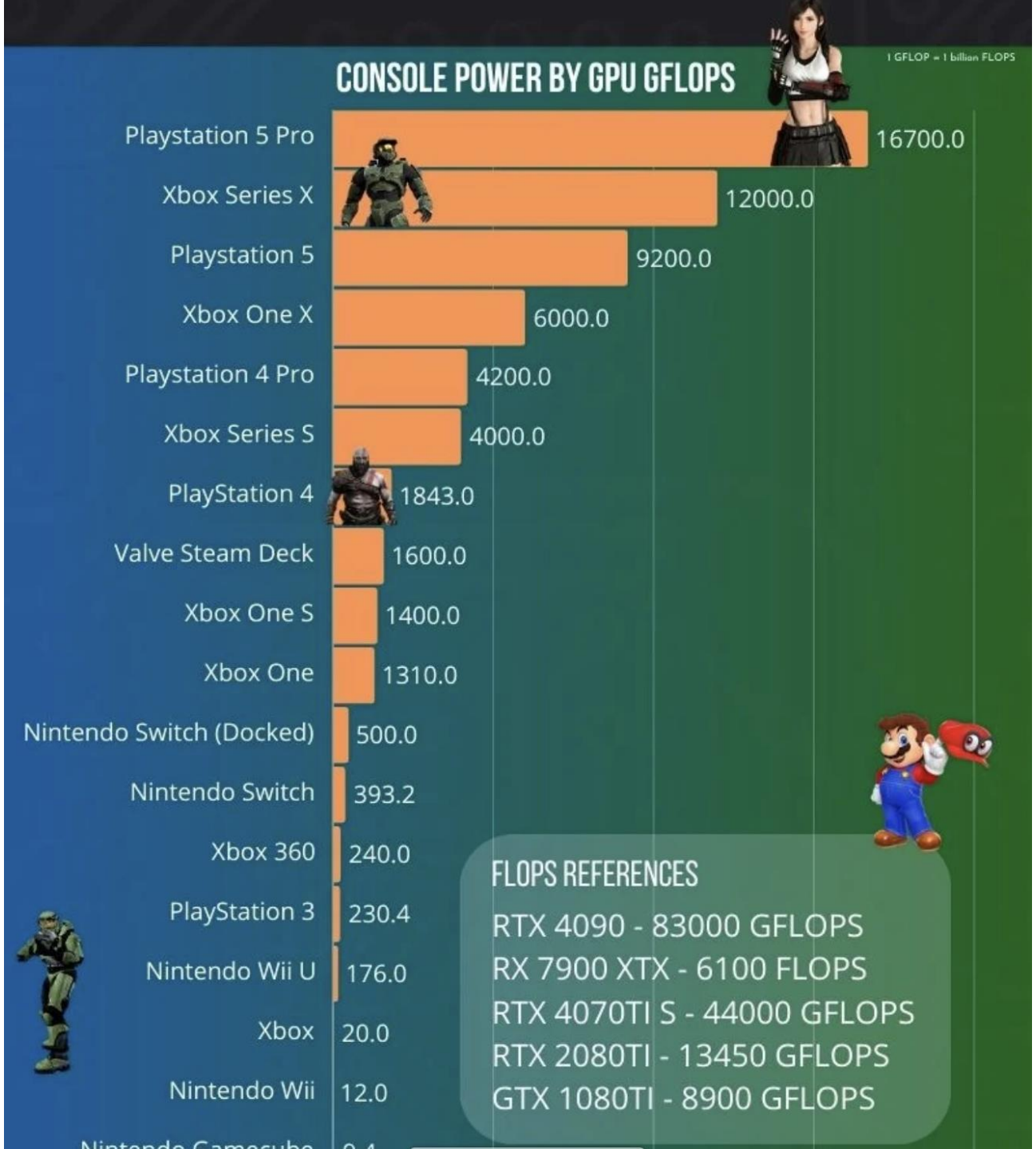
- How these systems and applications are **integrated** to solve problems?

From Supercomputers to the Cloud

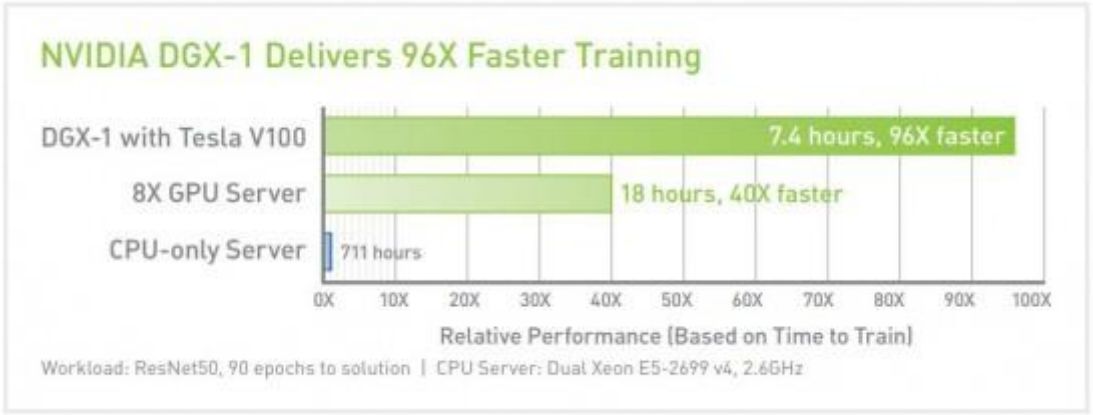


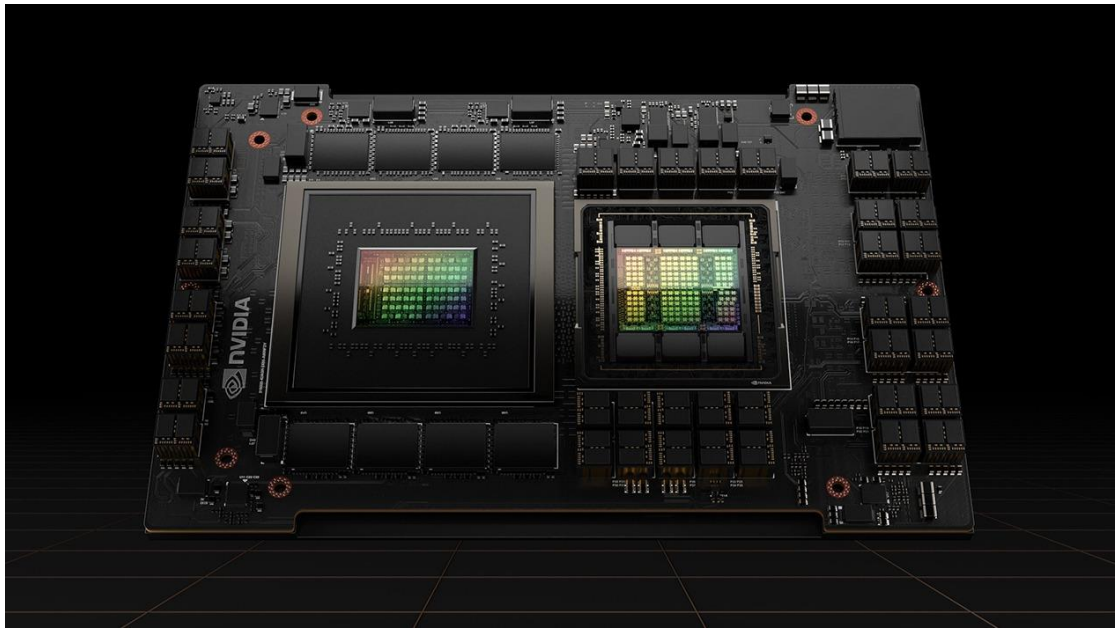
#	T	Site	Manufacturer	Computer	Country	HPCG [Pflop/s]	HPL [Pflop/s]	HPCG/ Peak	HPCG/ HPL
1	1	RIKEN-CCS	Fujitsu	Fugaku Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D	Japan	16.005	442.0	3.0%	3.6%
2	2	Oak Ridge National Laboratory	IBM	Summit IBM Power System, P9 22C 3.07 GHz, Volta GV100, EDR	USA	2.926	148.6	1.5%	2.0%
3	5	NERSC - Lawrence Berkeley National Laboratory	HPE	Perlmutter HPE Cray EX235n, AMD EPYC 64C 2.45GHz, NVIDIA A100, Slingshot-10	USA	1.905	64.6	2.0%	3.0%
4	3	Lawrence Livermore National Laboratory	IBM	Sierra IBM Power System, P9 22C 3.1 GHz, Volta GV100, EDR	USA	1.796	94.6	1.4%	1.9%
5	6	NVIDIA Corporation	NVIDIA	Selene DGX A100 SuperPOD, AMD 64C 2.25GHz, NVIDIA A100, Mellanox HDR	USA	1.623	63.5	2.1%	2.6%
6	8	Forschungszentrum Jülich (FZJ)	Atos	JUWELS Booster Module BullSequana XH2000, AMD EPYC 24C 2.8GHz, NVIDIA A100, Mell. HDR	Germany	1.275	44.1	1.8%	2.9%
7	11	Saudi Aramco	HPE	Dammam-7 Cray CS-Storm, Xeon 20C 2.5GHz, NVIDIA T. V100, IB HDR 100	Saudi Arabia	0.881	22.4	1.6%	3.9%
8	9	Eni S.p.A	Dell EMC	HPC5 PowerEdge C4140, Xeon 24C 2.1GHz, NVIDIA V100, Mellanox HDR	Italy	0.860	35.5	1.7%	2.4%
9	13	Information Technology Center, The University of Tokyo	Fujitsu	Wisteria/BDEC-01 (Odyssey) PRIMEHPC FX1000, A64FX 48C 2.2GHz, Tofu interconnect D	Japan	0.818	2	2%	3.7%
10	40	Japan Agency for Marine-Earth Science and Technology	NEC	Earth Simulator - SX-Aurora TSUBASA SX-Aurora TSUBASA A401-8, Vector Eng. Type20B 8C 1.6GHz, IB HDR 200	Japan	0.748	10	6%	7.5%

https://twitter.com/top500supercomp



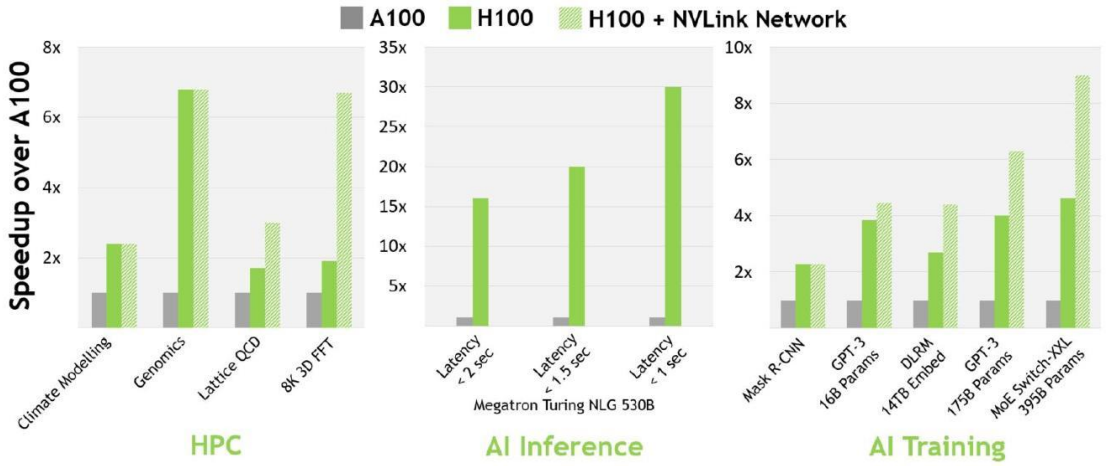
https://thegamingsetup.com/
console-power-comparison-chart


NVIDIA DGX-1 Delivers 96X Faster Training
DGX-1 with Tesla V100
7.4 hours, 96X faster
8X GPU Server
18 hours, 40X faster
CPU-only Server
711 hours
0X
10X
20X
30X
40X
50X
60X
70X
80X
90X
100X
Relative Performance (Based on Time to Train)
Workload: ResNet50, 90 epochs to solution | CPU Server: Dual Xeon E5-2699 v4, 2.6GHz


NVIDIA

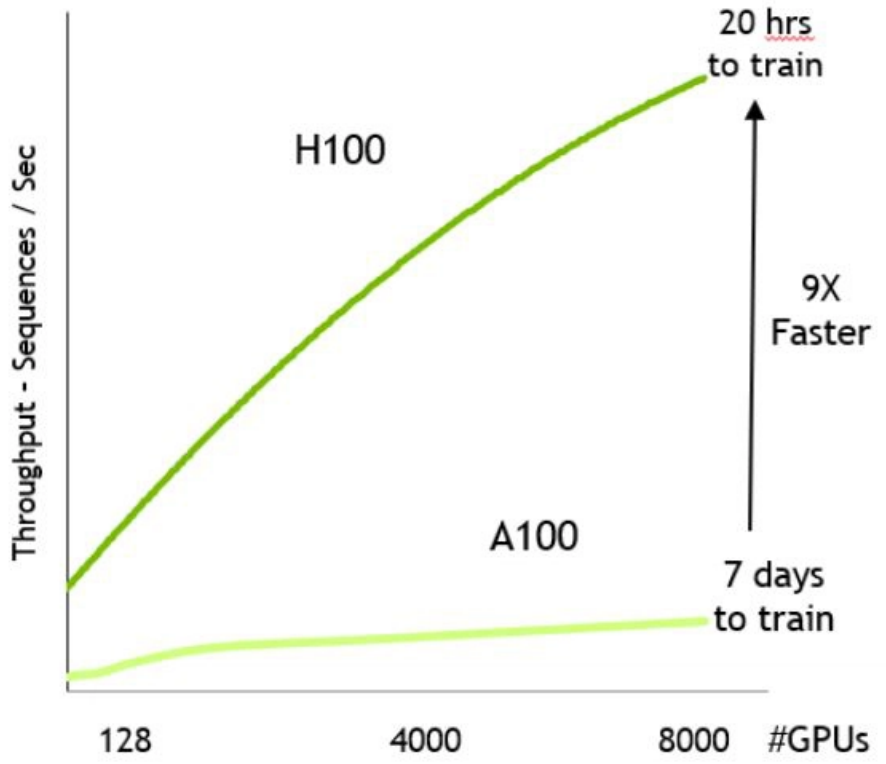
Form Factor	H100 SXM	H100 PCIe
FP64	34 teraFLOPS	26 teraFLOPS
FP64 Tensor Core	67 teraFLOPS	51 teraFLOPS
FP32	67 teraFLOPS	51 teraFLOPS
TF32 Tensor Core	989 teraFLOPS*	756teraFLOPS*
BFLOAT16 Tensor Core	1,979 teraFLOPS*	1,513 teraFLOPS*
FP16 Tensor Core	1,979 teraFLOPS*	1,513 teraFLOPS*
FP8 Tensor Core	3,958 teraFLOPS*	,026 teraFLOPS*
INT8 Tensor Core	3,958 TOPS*	3,026 TOPS*
GPU memory	80GB	80GB

~250 PS-5



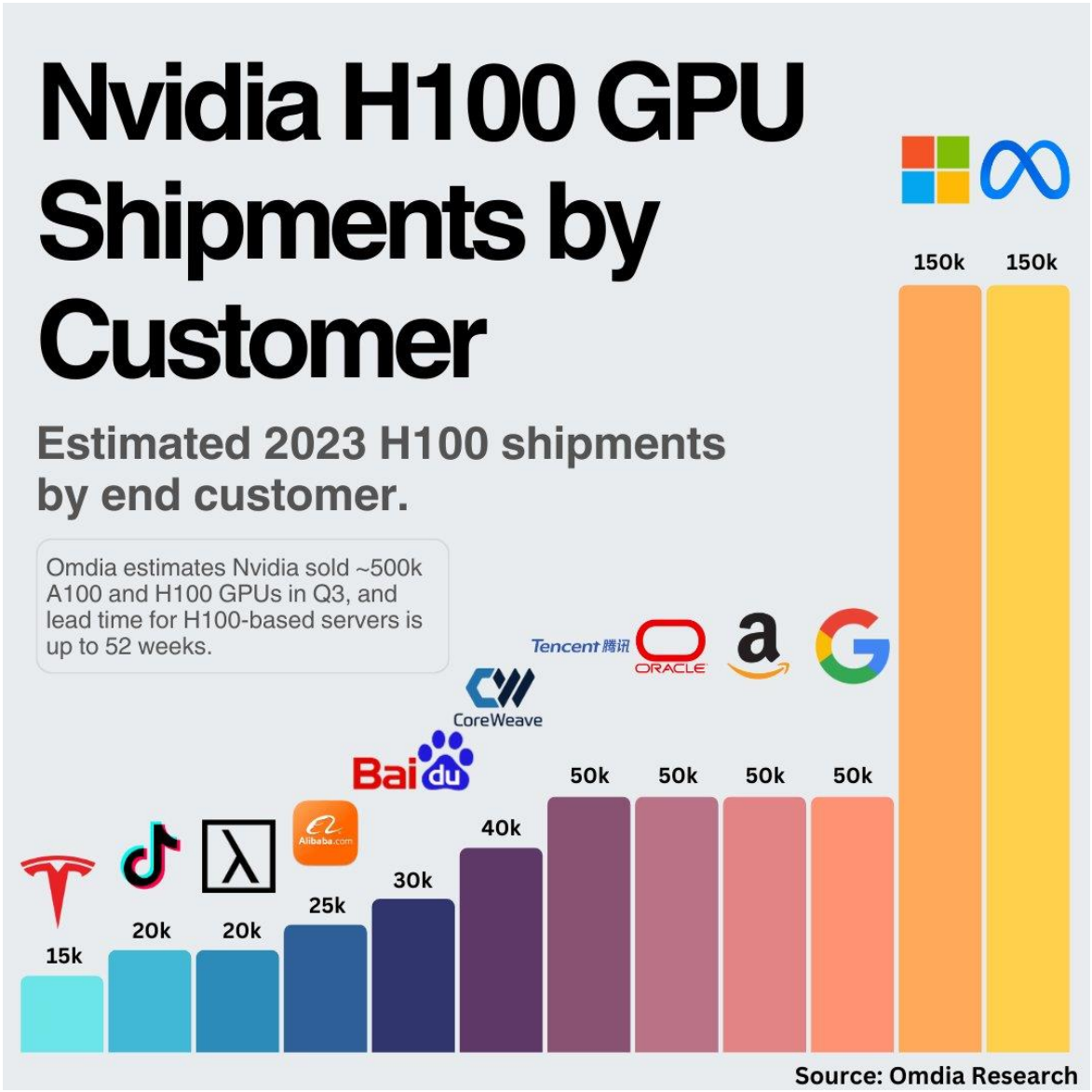
TRAINING

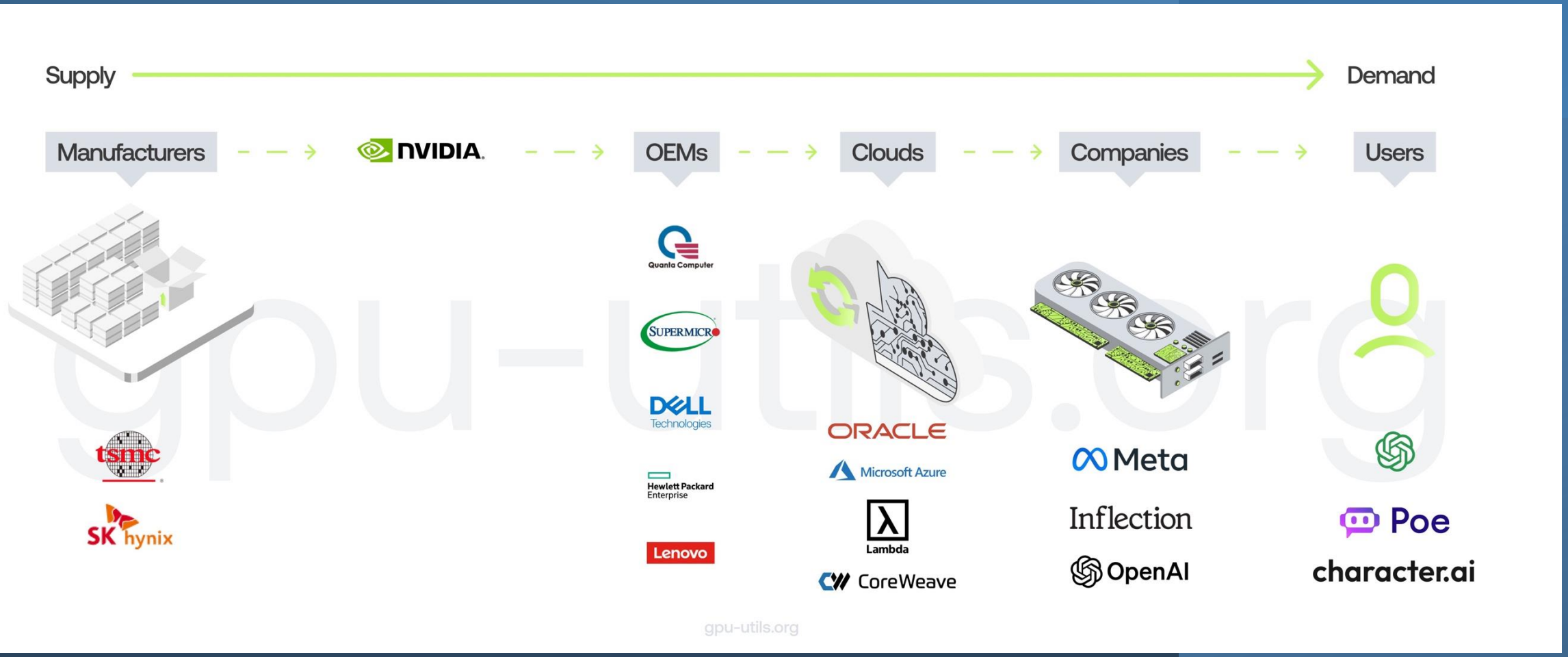
Up to 9X More Throughput



Mixture of Experts (395B) Training vs A100

Projected performance subject to change
Training Mixture of Experts (MoE) Transformer Switch-XXL variant with 395B parameters on 1T token dataset


Nvidia H100 GPU Shipments by Customer
Estimated 2023 H100 shipments by end customer.
Omdia estimates Nvidia sold ~500k A100 and H100 GPUs in Q3, and lead time for H100-based servers is up to 52 weeks.
15k
20k
20k
25k
30k
40k
50k
50k
50k
50k
150k
150k
Tencent 腾讯
ORACLE
CoreWeave
Baidu
Alibaba.com
Source: Omdia Research


Supply
Demand
Manufacturers
NVIDIA.
OEMs
Clouds
Companies
Users
Quanta Computer
SUPERMICRO
DELL Technologies
Hewlett Packard Enterprise
Lenovo
tsmc
SK hynix
ORACLE
Microsoft Azure
Lambda
CoreWeave
Meta
Inflection
OpenAI
Poe
character.ai
gpu-utils.org

The cloud: computation for the rest of us

Acceder a NVIDIA a través de nuestros partners de nube

The cloud: computation for the rest of us

NVIDIA V100	1 GPU	16 GB HBM2	USD 2.48 / 1 hour	USD 1.562 / 1 hour	USD 1.116 / 1 hour
	2 GPU	32 GB HBM2			
	4 GPU	64 GB HBM2			
	8 GPU	128 GB HBM2			
NVIDIA P100	1 GPU	16 GB HBM2	USD 1.46 / 1 hour	USD 0.919 / 1 hour	USD 0.657 / 1 hour
	2 GPU	32 GB HBM2			
	4 GPU	64 GB HBM2			

The cloud: computation for the rest of us

p5.48xlarge

The p5.48xlarge instance is in the gpu instance family with 192 vCPUs, 2048.0 GiB of memory and 3200 Gibps of bandwidth starting at $98.32 per hour.

Pricing

$98.320	$9.832	N/A	$43.157
On Demand	Spot	1 Yr Reserved	3 Yr Reserved

This instance is available in Capacity Blocks.

US East (N. Virginia) | Linux

Per Hour | No Upfront

Request a Free Vantage Demo

Learn more about how you can observe and

Instance Details

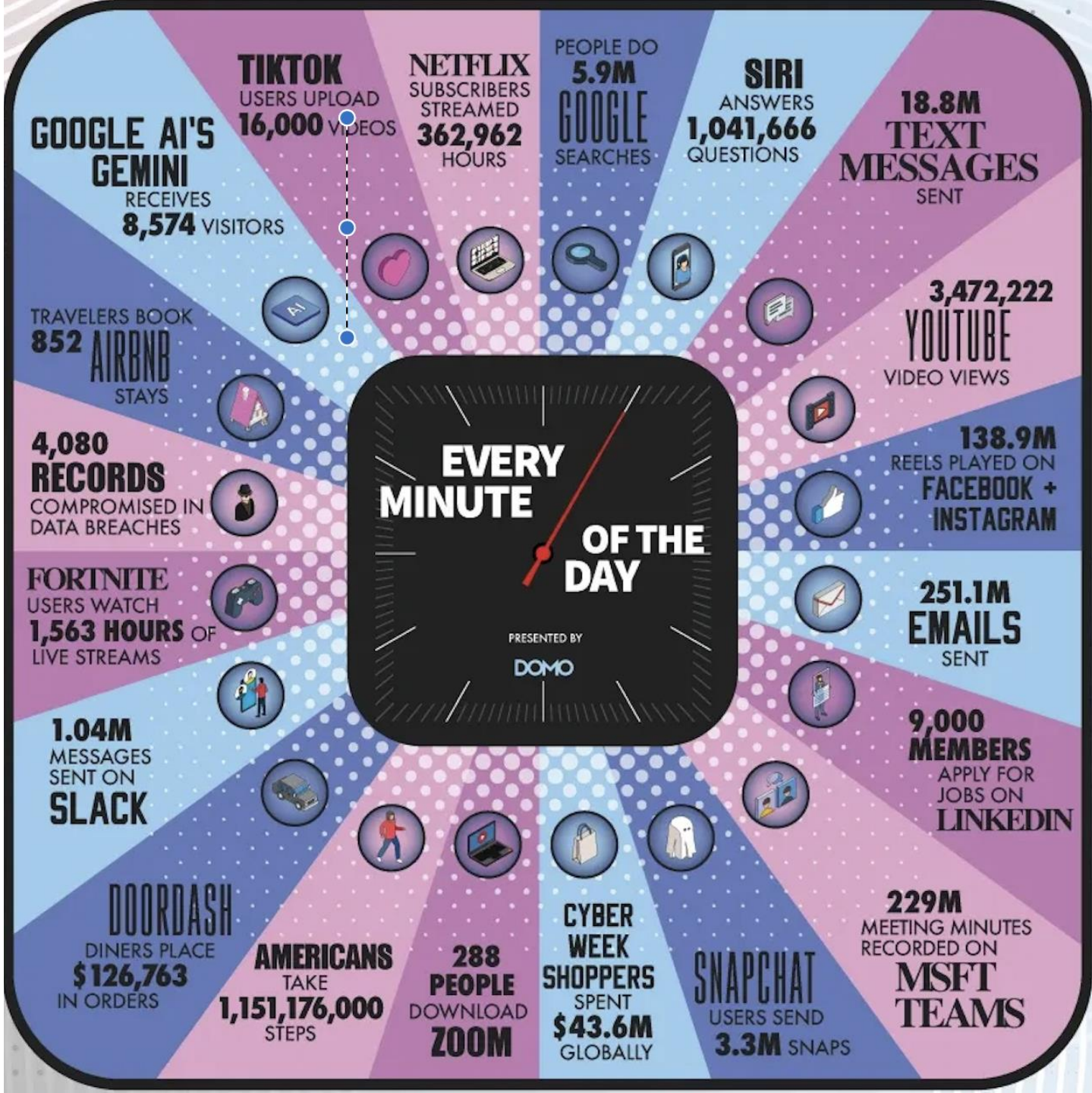
Compute	Value
vCPUs	192
Memory (GiB)	2048.0
Memory per vCPU (GiB)	10.67
Physical Processor	AMD EPYC 7R13 Processor
Clock Speed (GHz)	2.95
CPU Architecture	x86_64
GPU	8
GPU Architecture	nvidia h100
Video Memory (GiB)	640
GPU Compute Capability (?)	9.0
FPGA	0

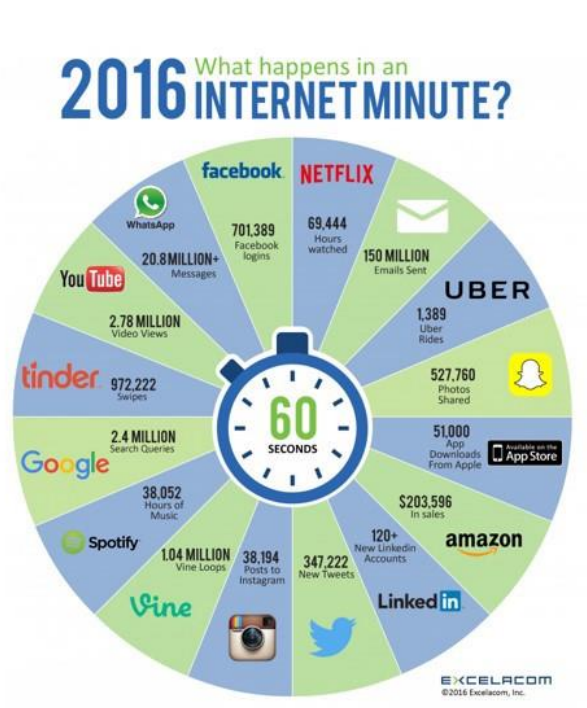
Data intensive applications

Introduction

Data intensive applications

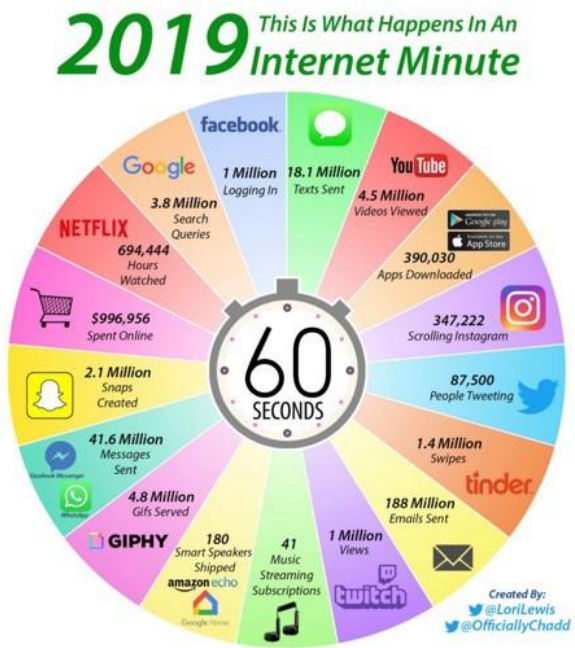
- Not limited by CPU power
- Relevant amount of data
- Complexity of data
- Speed of data change

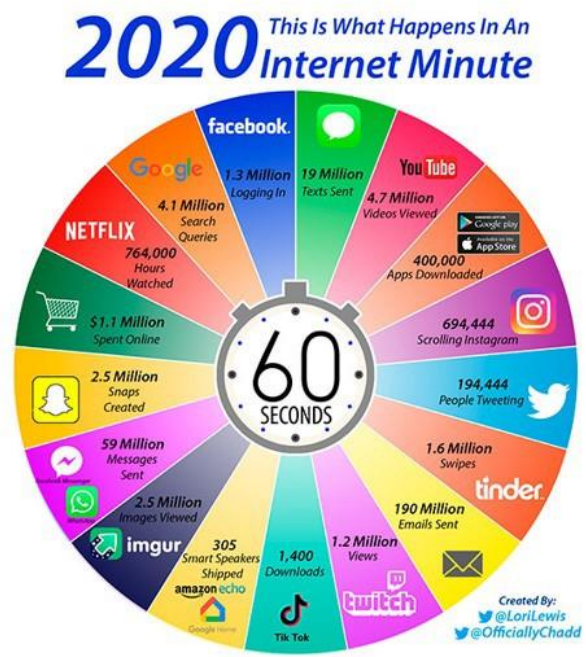



2016 What happens in an INTERNET MINUTE?
60 SECONDS
facebook
701,389 Facebook logins
NETFLIX
69,444 Hours watched
150 MILLION Emails Sent
UBER
1,389 Uber Rides
527,760 Photos Shared
51,000 App Downloads From Apple
$203,596 In sales
amazon
120+ New LinkedIn Accounts
Linked in
347,222 New Tweets
38,194 Posts to Instagram
1.04 MILLION Vine Loops
Vine
Spotify
38,052 Hours of Music
Google
2.4 MILLION Search Queries
tinder
972,222 Swipes
YouTube
2.78 MILLION Video Views
20.8 MILLION+ Messages
WhatsApp
EXCELACOM
©2016 Excelacom, Inc.


2017 This Is What Happens In An Internet Minute
60 SECONDS
facebook
900,000 Logins
16 Million Text Messages
YouTube
4.1 Million Videos Viewed
342,000 Apps Downloaded
46,200 Posts Uploaded
Instagram
452,000 Tweets Sent
990,000 Swipes
tinder
156 Million Emails Sent
40,000 Hours Listened
Spotify
50 Voice-First Devices Shipped
amazon echo
120 New Accounts Created
Linked in
15,000 GIFs Sent via Messenger
1.8 Million Snaps Created
$751,522 Spent Online
70,017 Hours Watched
NETFLIX
3.5 Million Search Queries
Google
Created By:
@LoriLewis
@OfficiallyChadd


2018 This Is What Happens In An Internet Minute
60 SECONDS
facebook
973,000 Logins
18 Million Text Messages
YouTube
4.3 Million Videos Viewed
375,000 Apps Downloaded
174,000 Scrolling Instagram
481,000 Tweets Sent
1.1 Million Swipes
tinder
187 Million Emails Sent
936,073 Views
twitch
67 Voice-First Devices Shipped
amazon echo
38 Million Messages
25,000 GIFs Sent via Messenger
2.4 Million Snaps Created
$862,823 Spent Online
266,000 Hours Watched
NETFLIX
3.7 Million Search Queries
Google
Created By:
@LoriLewis
@OfficiallyChadd


2019 This Is What Happens In An Internet Minute
60 SECONDS
facebook
1 Million Logging In
18.1 Million Texts Sent
YouTube
4.5 Million Videos Viewed
390,030 Apps Downloaded
347,222 Scrolling Instagram
87,500 People Tweeting
1.4 Million Swipes
tinder
188 Million Emails Sent
1 Million Views
twitch
41 Music Streaming Subscriptions
180 Smart Speakers Shipped
amazon echo
GIPHY
4.8 Million Gifs Served
41.6 Million Messages Sent
2.1 Million Snaps Created
$996,956 Spent Online
694,444 Hours Watched
NETFLIX
3.8 Million Search Queries
Google
Created By:
@LoriLewis
@OfficiallyChadd


2020 This Is What Happens In An Internet Minute
60 SECONDS
facebook
1.3 Million Logging In
19 Million Texts Sent
YouTube
4.7 Million Videos Viewed
400,000 Apps Downloaded
694,444 Scrolling Instagram
194,444 People Tweeting
1.6 Million Swipes
tinder
190 Million Emails Sent
1.2 Million Views
twitch
Tik Tok
1,400 Downloads
305 Smart Speakers Shipped
amazon echo
imgur
2.5 Million Images Viewed
59 Million Messages Sent
2.5 Million Snaps Created
$1.1 Million Spent Online
764,000 Hours Watched
NETFLIX
4.1 Million Search Queries
Google
Created By:
@LoriLewis
@OfficiallyChadd


2021 This Is What Happens In An Internet Minute
60 SECONDS
facebook
1.4 Million Scrolling
21.1 Million Texts Sent
YouTube
500 Hours Content Uploaded
414,764 Apps Downloaded
695,000 Stories Shared
200,000 People Tweeting
2 Million Swipes
tinder
197.6 Million Emails Sent
2 Million Views
twitch
Tik Tok
5,000 Downloads
932 Smart Audio Devices Shipped
amazon echo
imgur
3 Million Images Viewed
69 Million Messages Sent
3.4 Million Snaps Created
$1.6 Million Spent Online
28,000 Subscribers Watching
NETFLIX
9,132 Connections Made
Linked in
Google
Created By:
@LoriLewis
@OfficiallyChadd

What is a data system?

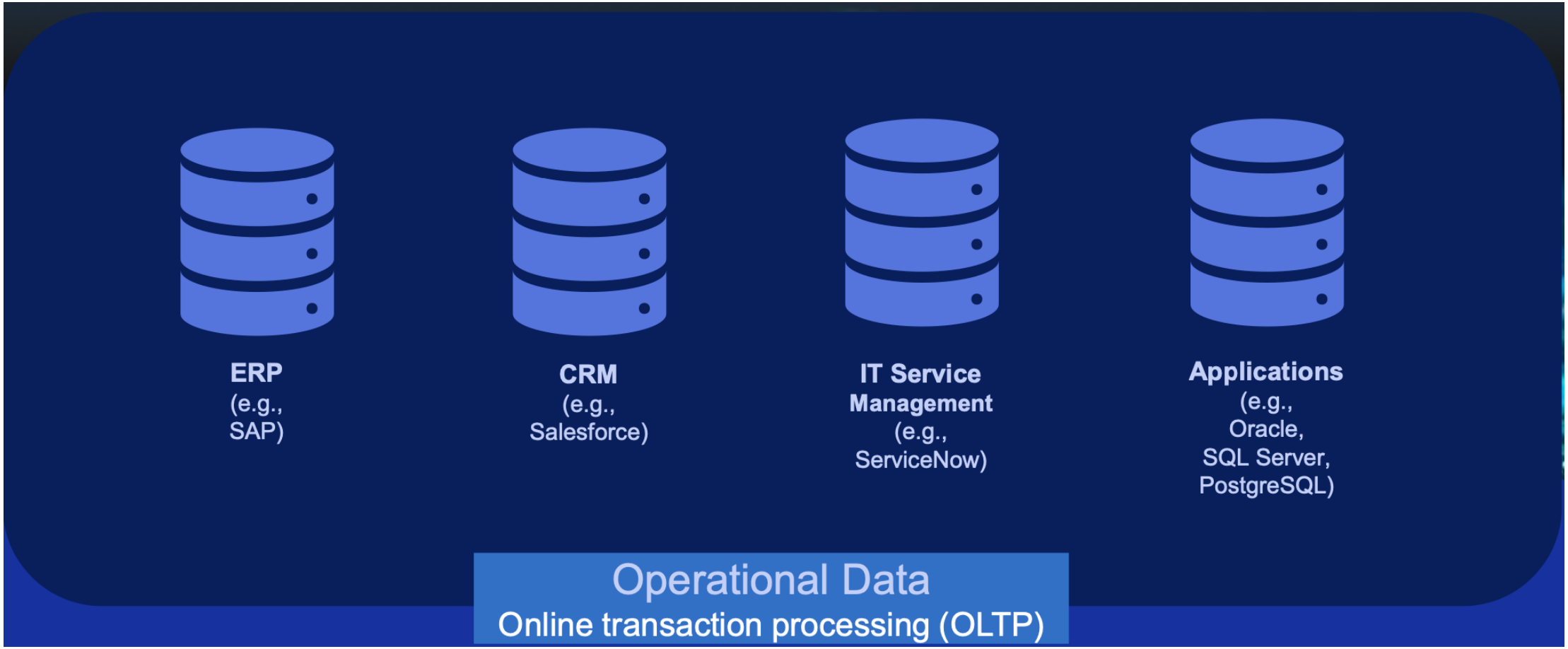
- New tools for data storage and processing
- New applications with wide range demands of data processing
- Not a single tool can meet data processing and storage needs

- Processing is divided into tasks for a single tool
- Different tools are linked with application code

Data intensive applications needs

- **Database**: store data for later processing
- **Cache**: remember the result of an expensive operation
- **Search index**: search data using keywords
- **Stream processing**: send messages to be handled asynchronously
- **Batch processing**: process a large amount of accumulated data

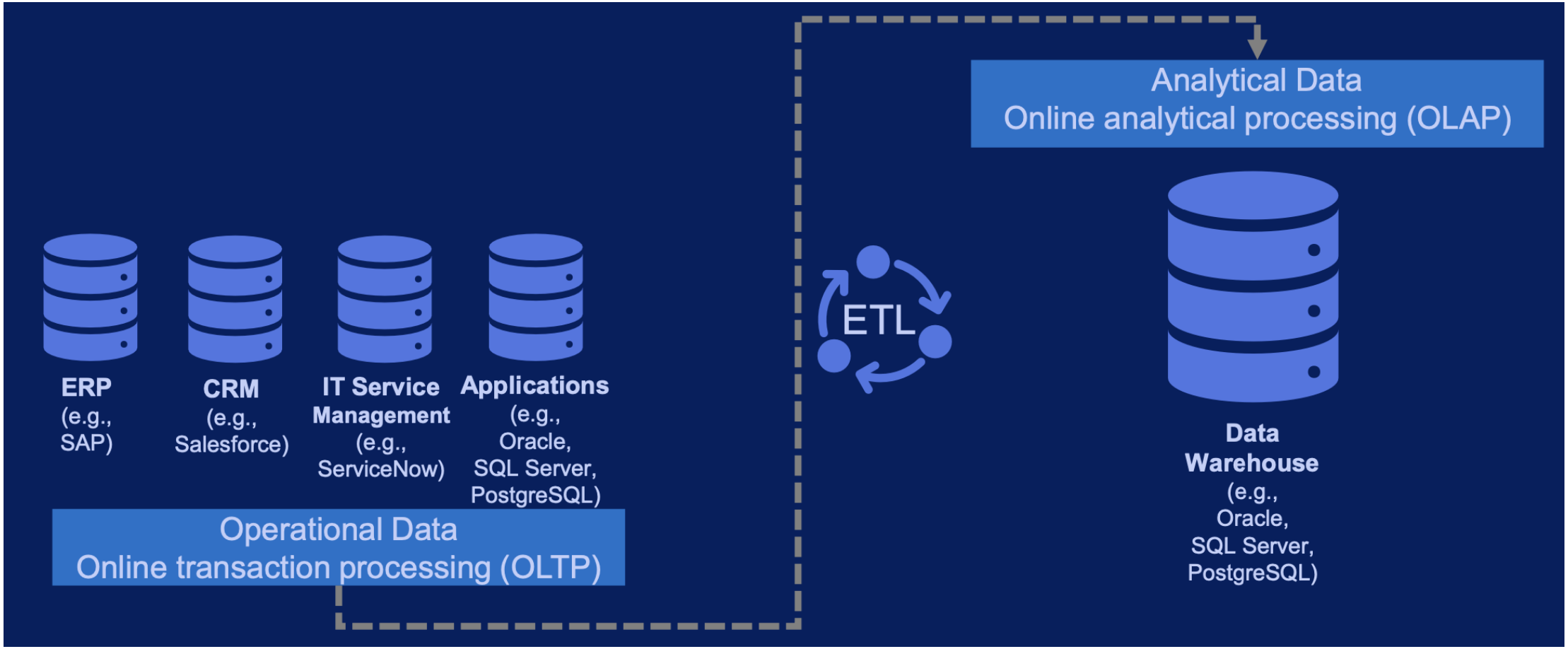
Sources of operational data



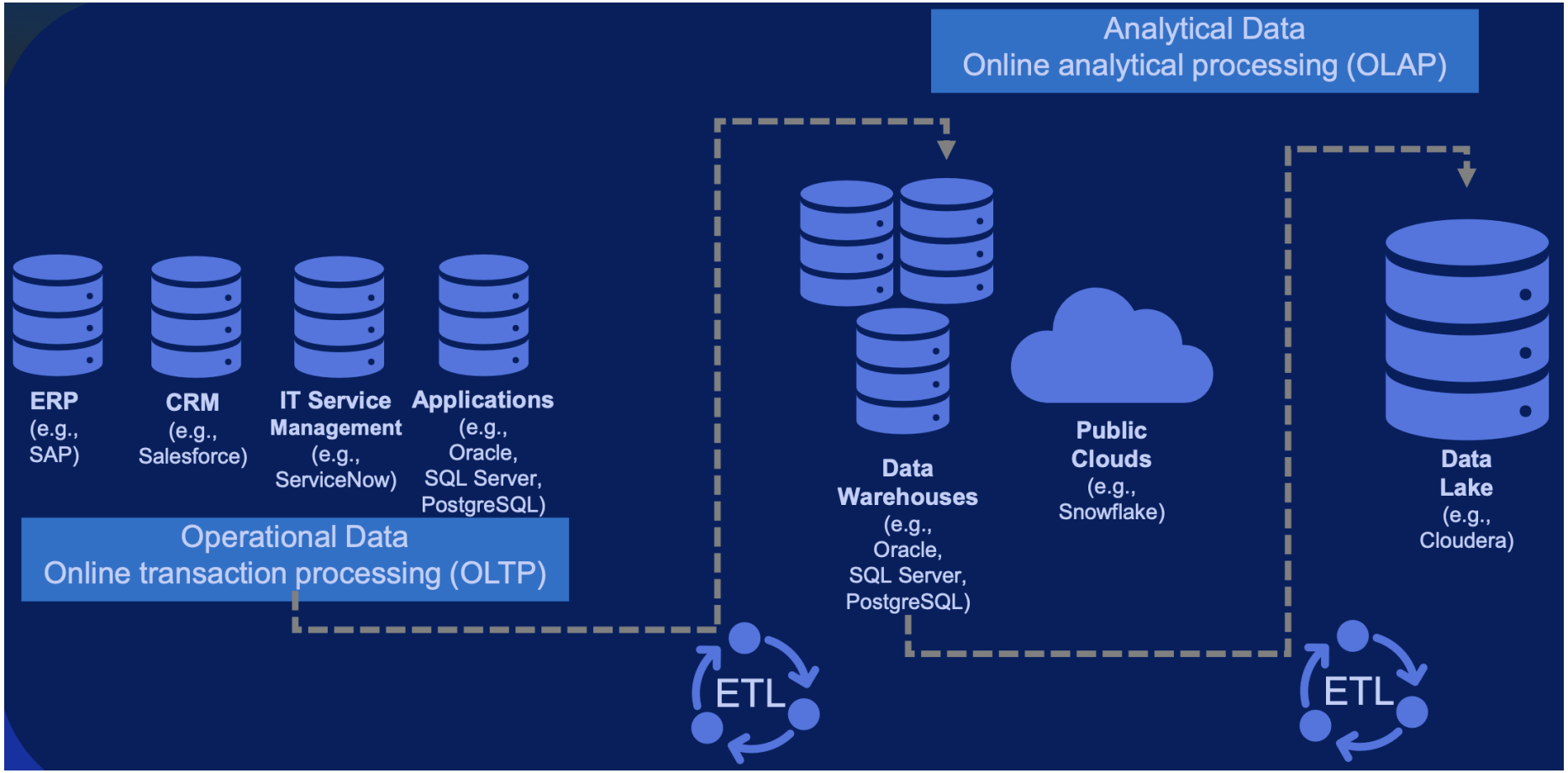
Modern tools for customer/provider mgmt

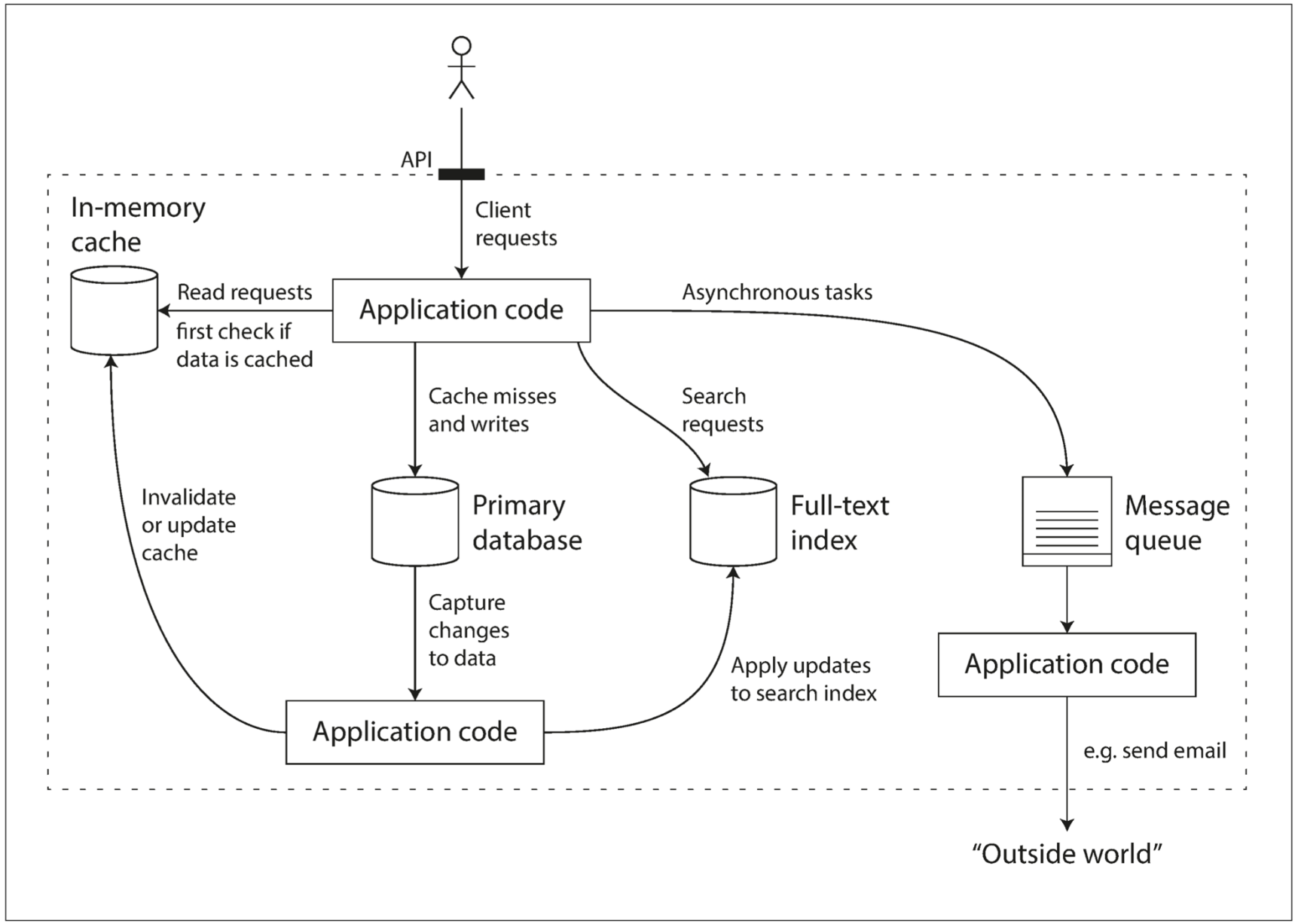
- **ERP: SAP, Microsoft Dynamics**
  - https://www.microsoft.com/es-es/dynamics-365#productdemos
  - https://www.sap.com/spain/products/erp/what-is-sap-erp.html
- **CRM: Salesforce, Odoo**
  - https://www.salesforce.com/mx/crm/
  - https://www.salesforce.com/es/customer-success-stories
  - https://www.odoo.com/es_ES/app/crm
- **Databases:**
  - E-commerce analytics
  - E-mail campaigns
  - Logistics data

Next step: data integration for analytics



Data warehousing for OLAP





Data Management System

Applications as a service

- Use small general-purpose tools as components
- Build a special-purpose data system
- You are now a data system designer!
- Your system will available to programmers via API

Things to do with a data system

- Ensure data remains correct and complete
- Provide good performance
- **Be able to scale when load increases**
- Provide good API for the service

Three important concerns

- **Reliability**: system should work correctly when errors happen
- **Scalability**: how to deal with growth (data, traffic, complexity)
- **Maintainability**: maintaining and improving behaviour should be a productive task

Reliability:
Working correctly when things go wrong

- Faults: component starts working under spec
- Failures: system stops providing a required service
- Fault-tolerance mechanisms: prevent faults to produce failures
- netflix chaos monkey
  - https://netflix.github.io/chaosmonkey/

subsea datacenters powered by offshore renewable energy

https://natick.research.microsoft.com/ https://www.youtube.com/watch?v=lBeepqQBpvU

Scalability

- System ability to cope with increased load
- How can added computational resources cope with growth?

- How to describe the load of a system?
- Depends on the architecture of the system
  - Requests per second to web server
  - Reads/writes request ratio to a data base
  - Simultaneous number of users in videoconference

Twitter post and timeline

- **POST tweet**: user can publish a new message to their followers
  - insert the new tweet into a global collection of tweets
- **Home Timeline**: user can view messages posted by people they follow
  - look up all the people I follow
  - find all the tweets for each of those users
  - merge all published tweets (sorted by time)

Post and timeline load metrics

POST tweet: user can publish a new message to their followers

- 4.600 requests per second on average
- 12.000 requests per second at peak

Home Timeline: user can view tweets posted by people they follow

- 300.000 requests per second

- Which operation is more common?

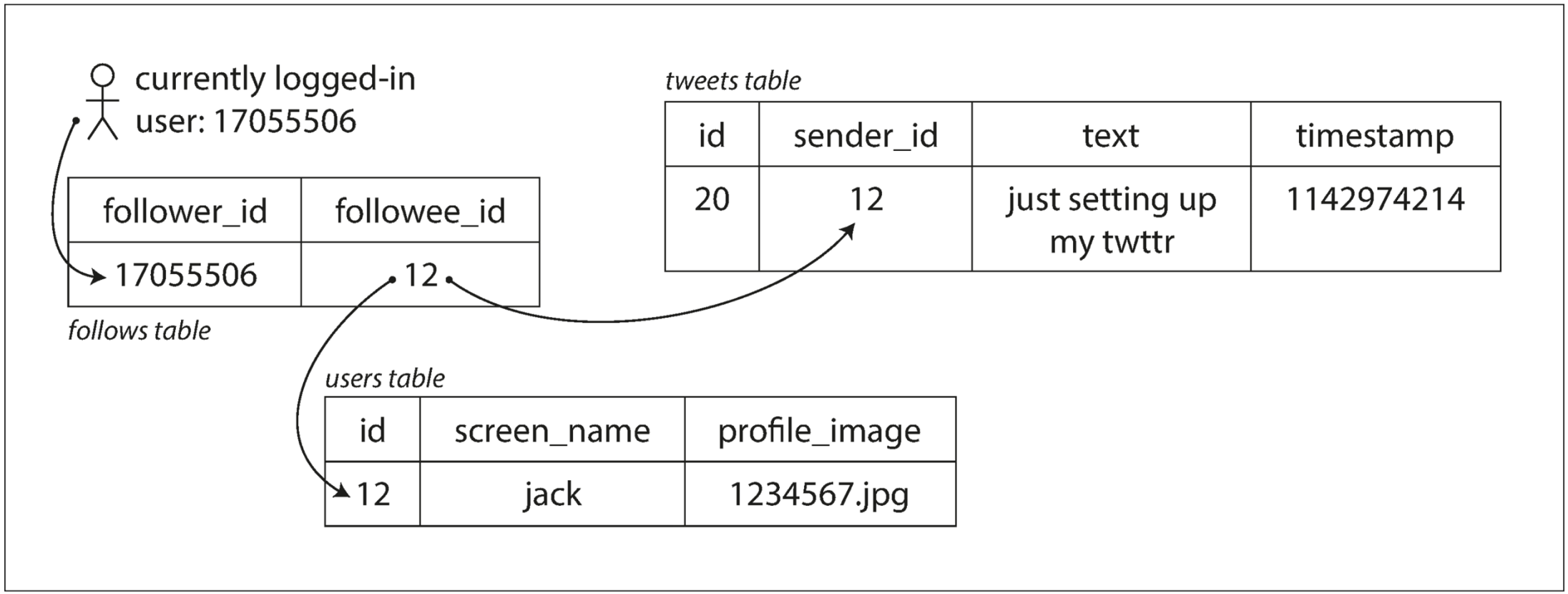
Option 1: global collection of tweets

- Insert each tweet into a global collection

When user requests home timeline:

1. Look up all users he/she follows
2. Find all the tweets of all these users
3. Sort all tweets by time

Option 1: global collection of tweets



Option 1: global collection of tweets

```
SELECT tweets.*
FROM tweets, follows
WHERE follows.followee_id = tweets.sender_id AND
        follows.follower_id = current_user
```

Main problem: load of timeline queries

- For each N user (follower_id)
  - Get all M tweets from tweets table

- Problem: N*M messages to read

Twitter scaling challenge: fan out

- Each user follows N other users
- Each user is followed by M other users
- Potentially an N*M number of messages to manage

- How to deal with this quadratic growth?

https://www.infoq.com/presentations/Twitter-Timeline-Scalability

Main problem: load of timeline queries

- For each N user (follower_id)
  - Get all M twits from twits table

Problem: N*M messages to read

- FACT: **average rate of published tweets** is **100 times lower** than rate of home timeline reads => read is x100 more common that write
- IDEA: **optimize the most common case**.
  - Try to reduce the cost of reading tweets

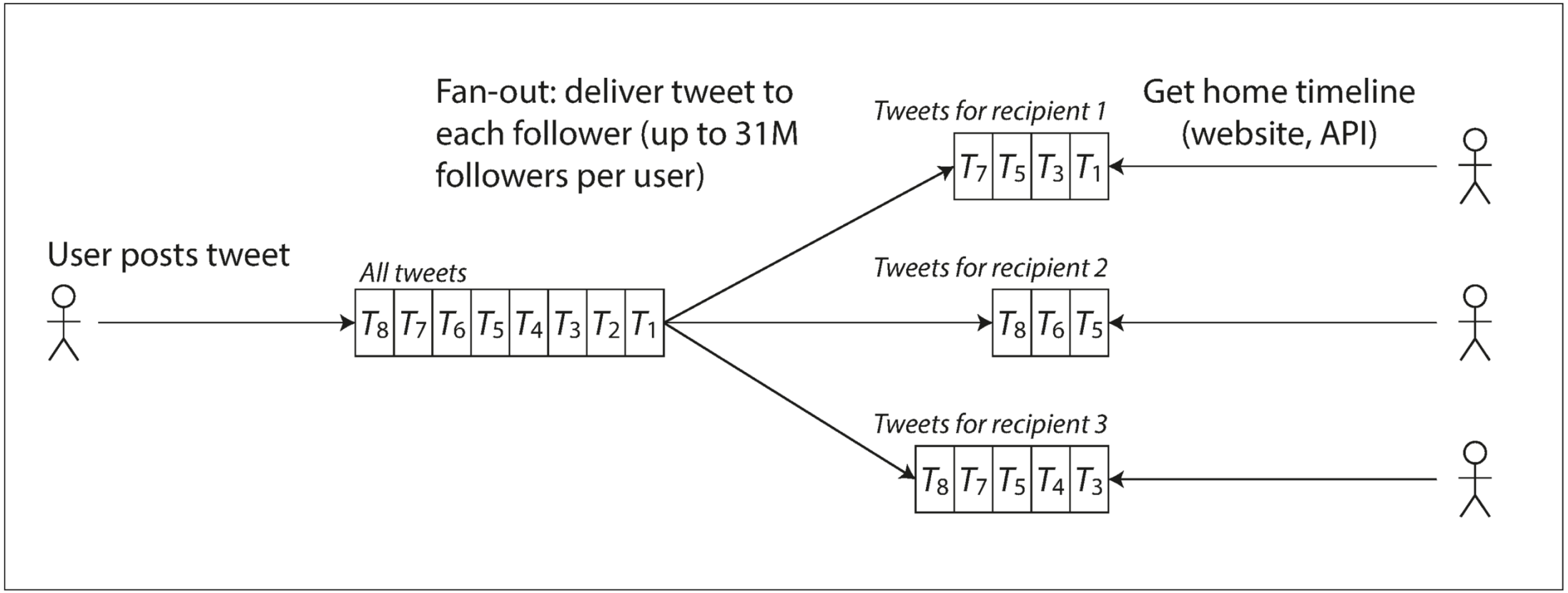
Option 2: user timeline cache

- Maintain a cache of each user's timeline
- Like a mailbox of tweets for each recipient user

When a **user posts a tweet**:

1. Look up all the people who follow that user
2. Insert the new tweet into each of followers home timeline caches

Option 2: user timeline cache



Option 2: user timeline cache

When a user posts a tweet

1. Look up all the people who follow that user
2. Insert the new tweet into each of their home timeline caches

When a user wants its timeline

- Read its local cache of tweets

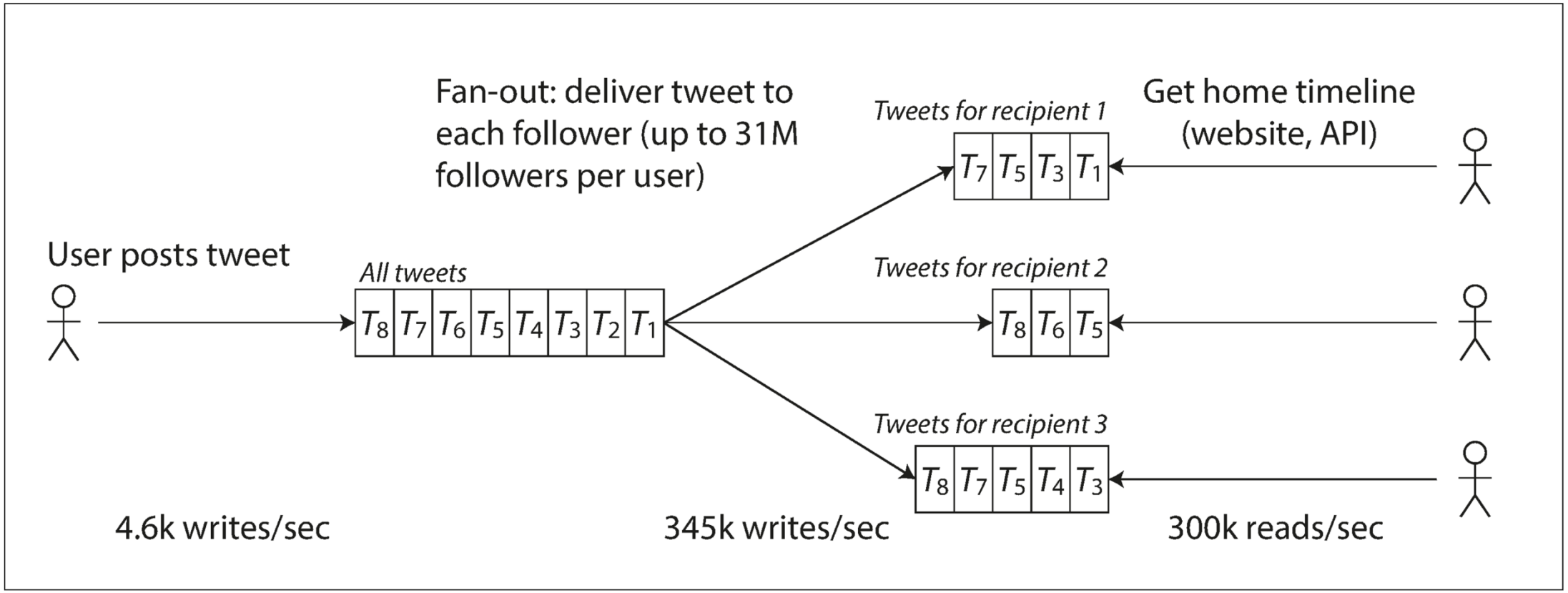
- **Timeline read now is easier, as it is already computed!**

Rough load rates of option 2

Posting a tweet now requires extra work

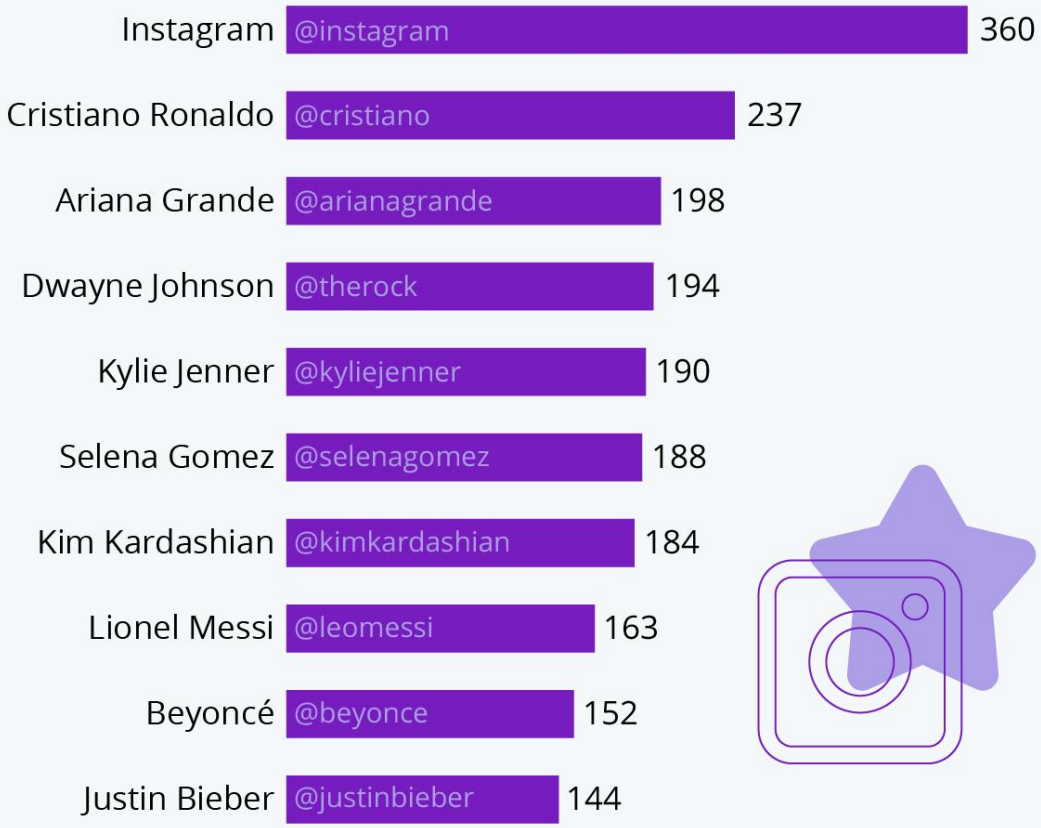
- Average tweet is delivered to 75 followers
- 4.600 tweets per second
- (x75) **345.000 writes per second** to home timeline caches

Option 2: user timeline cache



Las cuentas de Instagram con más seguidores

Perfiles con mayor número de seguidores en Instagram en agosto de 2020 (en millones)



Datos del 20 de agosto de 2020.
Fuente: Trackalytics

statista

Celebrities management?

Celebrities latency challenge

- FACT 1: Twitter delivers messages to timelines in **5 seconds**
- FACT 2: some users have **>100 million followers**!

- **A single twit generates 100 million writes to home timelines**
- **how to process that in 5 seconds?**

Celebrities challenge

- FACT 1: Twitter delivers messages to timelines in **5 seconds**
- FACT 2: some users have **100 million followers**!
- **A single tweet generates 300 million writes to home timelines to be processed in 5 seconds**

- Distribution of followers per user?
- Weight of user tweet frequency?

Hybrid approach to timeline computation

- Most users get their timeline preprocessed with caches
- Celebrity messages are fetched and merged when reading home timeline
- When requested, celebrity tweets are merged with user home timeline following approach 1

Load requirements of your project?

Once you have described the load on your system, you can investigate what happens when the load increases

- When you increase a load parameter and keep the system resources (CPU, memory, network bandwidth, etc.) unchanged, how is the performance of your system affected?
- When you increase a load parameter, how much do you need to increase the resources if you want to keep performance unchanged?

System performance

System performance

- How is performance affected when increasing load parameter in the same resources (CPU, memory, network bandwidth).
- When increasing a load parameter, how much more resources we need to keep performance unchanged?

Performance

- How to describe performance?
- Throughput:
  - number of records to process per second
  - time to process a job on a dataset
- Response time:
  - Time between a client sending a request and receiving a response
- Latency:
  - Duration that a request is waiting to be handled (request is latent waiting for service)

Coping with load

- Scale-up / vertical scaling: move to a more powerful server
  - Expensive
  - Limitations of technology
- Scale-out / horizontal scaling: distribute load across multiple smaller machines
  - Hard to maintain unified view/stateful data
- Elastic systems: add resources when load increases
  - Good for unpredictable events

Highly specific application decisions

Very different scenarios to manage

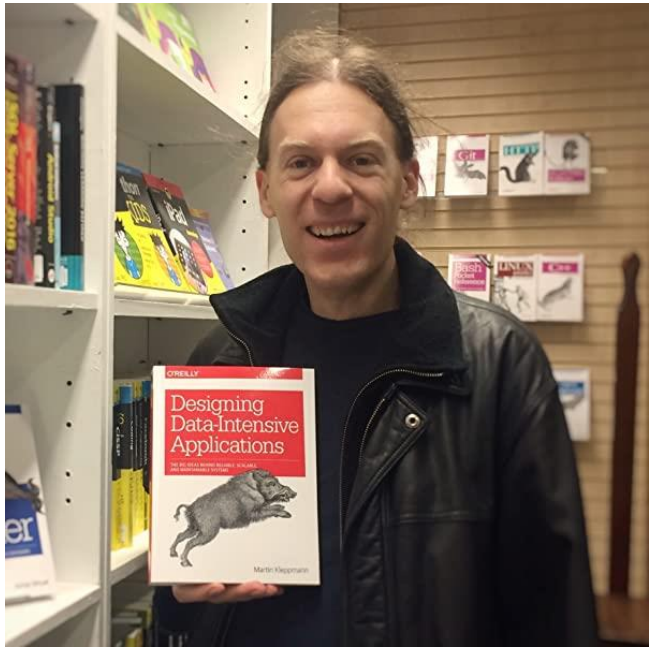
- 100.000 requests per second, 1KB per request
- 3 requests per minute, 2 GB per request

- Must care for common cases in load parameters
- Highly specific architecture decisions
- Made from general purpose building blocks

Chapter 1: “Designing Data Intensive Applications”

MARTIN KLEPPMAN – O’REILLY – 2017

https://dataintensive.net/



https://martin.kleppmann.com/

https://www.youtube.com/watch?v=iBeVXrOmwoQ

available at UAB digital library

https://bibcercador.uab.cat/permalink/34CSUC_UAB/1eqfv2p/alma991000617399706709