

SESIÓN 1: HDFS - EL EJÉRCITO DE COMPUTADORAS

Slides de Teoría - 15 minutos (12 slides × 1.2 min c/u)

CONCEPTOS FUNDAMENTALES QUE DEBEN SABER

Pre-requisitos de Sesión 0:

- Sistemas distribuidos básicos
- Docker containers funcionando
- Concepto de cluster
- Data locality principle

Conceptos Nuevos HDFS:

- **NameNode:** Cerebro del sistema (metadatos)
 - **DataNode:** Músculos del sistema (almacenamiento real)
 - **Blocks:** Cómo se divide un archivo grande
 - **Replication:** Copias automáticas para seguridad
 - **Metadata:** Información SOBRE los datos
 - **Fault Tolerance:** Qué pasa cuando algo falla
 - **Load Balancing:** Distribución inteligente de datos
-

SLIDE 1: EL DESAFÍO - "El Problema Imposible de Google"

Imagen de Referencia: Google en 2003 con servidores sobrecargados y datos creciendo exponencialmente

TÍTULO: "🎯 El Desafío que Parecía Imposible de Resolver"

EL DESAFÍO ÉPICO: "📈 PROBLEMA EXPONENCIAL: Google 2003"

- Internet crecía MÁS RÁPIDO que su capacidad de almacenarlo
- 3 mil millones de páginas web (y creciendo)
- Opciones: \$50M supercomputadora vs ¿????"

PORQUÉ IMPORTABA: "💡 **BREAKTHROUGH MOMENT:** Si Google no resolvía esto:

- No habría buscador universal
- No habría Android, Gmail, YouTube
- El internet sería fragmentado y lento"

CÓMO NACIÓ LA REVOLUCIÓN: "🧠 **INSIGHT GENIAL:** ¿Y si 1000 computadoras baratas trabajaran como una gigante? Inventaron Google File System (GFS)"

EVOLUCIÓN: "🌟 **RESULTADO:** Hadoop HDFS = versión open source de GFS Ahora CUALQUIERA puede usar la tecnología de Google"

👉 **MENSAJE DE ÁNIMO:** "¡Hoy VAS A CONSTRUIR tu propia versión del sistema que revolucionó internet! ¡Te convertirás en arquitecto de sistemas como los genios de Google! 🚀"

SLIDE 2: ¿QUÉ ES HDFS?

Imagen de Referencia: Diagrama conceptual de sistema de archivos distribuido

TÍTULO: "HDFS: Hadoop Distributed File System"

ANALOGÍA SIMPLE: "Como Google Drive, pero en MUCHAS computadoras"

COMPARISON TABLE:

Tu Computadora	Google Drive	HDFS
1 disco duro	Servidores de Google	TU cluster de servidores
Si falla = pierdes todo	Google se encarga	Automáticamente se repara
Máximo ~1TB	Prácticamente ilimitado	Prácticamente ilimitado

KEY DIFFERENCES:

- **Escalabilidad:** Agrega más servidores = más espacio
- **Confiabilidad:** Si 1 servidor falla, datos siguen disponibles
- **Performance:** Múltiples servidores leyendo en paralelo
- **Control:** TÚ decides cómo distribuir los datos

REAL USAGE: "Netflix, Facebook, LinkedIn usan HDFS para petabytes de datos"

SLIDE 3: ARQUITECTURA HDFS - MASTER/SLAVE

Imagen de Referencia: Diagrama claro de NameNode y DataNodes con sus roles

TÍTULO: "Arquitectura HDFS: El Cerebro y los Músculos"

VISUAL HIERARCHY:

```
..... NAMENODE (Master)
..... "El Bibliotecario"
..... |
..... +-----+-----+
..... | ..... | ..... |
DATANODE1 DATANODE2 DATANODE3
"Estantes" "Estantes" "Estantes"
```

NAMENODE - EL CEREBRO:

- Guarda METADATOS (información sobre dónde están los archivos)
- NO guarda los archivos reales
- Como bibliotecario que sabe dónde está cada libro
- Si falla = problema grave (por eso tiene backups)

DATANODES - LOS MÚSCULOS:

- Guardan los DATOS reales en pedazos (blocks)
- Reportan su estado al NameNode cada 3 segundos
- Como estantes que guardan los libros
- Si uno falla = otros tienen copias

COMUNICACIÓN:

- Cliente pregunta a NameNode: "¿Dónde está archivo X?"
- NameNode responde: "Ve a DataNode 2 y 5"
- Cliente va directo a DataNodes por los datos

SLIDE 4: BLOCKS - DIVIDIR PARA CONQUISTAR

Imagen de Referencia: Archivo grande dividiéndose en bloques con distribución

TÍTULO: "Blocks: Cómo HDFS Divide Archivos Grandes"

VISUAL PROCESS:

Archivo Original (1GB)

..... ↓

[Block 1: 128MB] [Block 2: 128MB] [Block 3: 128MB] ... [Block 8: 128MB]

..... ↓

Distribución Automática:

DataNode1: Block 1, Block 4, Block 7

DataNode2: Block 2, Block 5, Block 8

DataNode3: Block 3, Block 6, Block 1 (copia)

¿POR QUÉ 128MB?:

- Suficientemente grande: Minimiza overhead de metadatos
- Suficientemente pequeño: Permite buena distribución
- Configurable: Puedes cambiarlo según necesidades

BENEFITS:

- **Paralelismo:** Múltiples DataNodes leen diferentes blocks simultáneamente
- **Distribución:** Un archivo puede estar en múltiples servidores
- **Flexibilidad:** Archivos grandes no limitados por tamaño de un disco

ANALOGY: "Como dividir una pizza grande en 8 pedazos para que 8 personas coman al mismo tiempo"

SLIDE 5: REPLICATION - COPIES PARA SEGURIDAD

Imagen de Referencia: Diagrama de bloques replicados en diferentes nodos

TÍTULO: "Replication: Copies Automáticas para Nunca Perder Datos"

DEFAULT REPLICATION: Factor 3 (cada block en 3 DataNodes diferentes)

VISUAL REPLICATION STRATEGY:

Block Original → DataNode 1 (Rack A)

..... ↓

Copy 1 → DataNode 2 (Rack A) - misma ubicación física

..... ↓

Copy 2 → DataNode 5 (Rack B) - diferente ubicación física

RACK AWARENESS:

- HDFS conoce la topología física de tu cluster
- Primera copia: Mismo rack (rápido)
- Segunda copia: Diferente rack (seguro)

SCENARIOS:

- **DataNode falla:** Otros 2 tienen copia
- **Rack completo falla:** Rack diferente tiene copia
- **Red lenta:** Lee desde el DataNode más cercano

TRADE-OFFS:

- Más replicación = Más seguridad, más espacio usado
- Menos replicación = Ahorro espacio, más riesgo
- Configurable por archivo según importancia

ENTERPRISE USAGE: "Bancos usan factor 5, startups usan factor 2"

SLIDE 6: HITO 1 - DOMINAS METADATA

Imagen de Referencia: NameNode como cerebro brillante controlando todo

TÍTULO: "🎯 HITO 1: Entiendes el Secreto Más Importante"

EL DESAFÍO: "😬 **CONCEPTO AVANZADO:** Metadata es MÁS importante que los datos mismos (Esto confunde al 70% de desarrolladores)"

PORQUÉ ES REVOLUCIONARIO: "💎 **INSIGHT CRUCIAL:** NameNode NO toca los datos, solo sabe dónde están Como bibliotecario que memoriza la ubicación de 10 millones de libros"

CÓMO FUNCIONA LA MAGIA: "🧠 **ARQUITECTURA GENIAL:**

- Metadatos en memoria → Acceso ultra-rápido
- Datos en discos → Almacenamiento masivo
- Separación perfecta de responsabilidades"

EVOLUCIÓN DE TU ENTENDIMIENTO: "🌟 **NIVEL DESBLOQUEADO:** Ahora entiendes por qué HDFS escala infinitamente"

💪 **MENSAJE DE ÁNIMO:** "¡EXCELENTE! Acabas de entender un concepto que toma años dominar. Ya piensas como arquitecto de sistemas distribuidos. ¡Este conocimiento vale oro en el mercado laboral! 💰
🌟"

SLIDE 7: WRITE PROCESS - CÓMO SE GUARDA UN ARCHIVO

Imagen de Referencia: Flujo paso a paso del proceso de escritura

TÍTULO: "Write Process: El Viaje de un Archivo a HDFS"

STEP-BY-STEP PROCESS:

```
1. Cliente: "Quiero guardar archivo.csv"
... ↓
2. NameNode: "Ok, usa DataNode1, DataNode3, DataNode7"
... ↓
3. Cliente → DataNode1: Envía datos
... ↓
4. DataNode1 → DataNode3: Replica datos
... ↓
5. DataNode3 → DataNode7: Replica datos
... ↓
6. DataNode7 → DataNode3: "Listo"
... ↓
7. DataNode3 → DataNode1: "Listo"
... ↓
8. DataNode1 → Cliente: "Archivo guardado"
... ↓
9. Cliente → NameNode: "Confirma ubicaciones"
```

IMPORTANT DETAILS:

- **Pipeline:** Datos fluyen en cadena para eficiencia
- **Acknowledgment:** Cada nodo confirma recepción
- **Checksums:** Verificación de integridad en cada paso
- **Atomic:** Todo o nada (no archivos parcialmente guardados)

FAILURE HANDLING: Si DataNode falla durante escritura, automáticamente usa otro

SLIDE 8: READ PROCESS - CÓMO SE LEE UN ARCHIVO

Imagen de Referencia: Flujo de lectura con múltiples DataNodes en paralelo

TÍTULO: "Read Process: Lectura Paralela Ultra-Rápida"

PARALLEL READ PROCESS:

1. Cliente: "Quiero leer archivo.csv"

... ↓

2. NameNode: "Block 1→DataNode1, Block 2→DataNode3, Block 3→DataNode5"

... ↓

3. Cliente lee EN PARALELO:

... |— Thread 1: Block 1 desde DataNode1

... |— Thread 2: Block 2 desde DataNode3

... |— Thread 3: Block 3 desde DataNode5

... ↓

4. Cliente ensambla blocks en orden correcto

SPEED ADVANTAGES:

- **Parallel I/O:** Múltiples discos leyendo simultáneamente
- **Network Distribution:** Tráfico distribuido entre DataNodes
- **Locality:** Lee desde DataNode más cercano
- **Caching:** DataNodes cachean blocks frecuentemente accedidos

SMART OPTIMIZATIONS:

- Si DataNode está ocupado → automáticamente usa réplica
- Si DataNode está lejos → usa réplica más cercana
- Si detecta corrupción → automáticamente usa otra réplica

RESULT: "10x-100x más rápido que un solo disco"

SLIDE 9: HITO 2 - FAULT TOLERANCE MASTER

Imagen de Referencia: Sistema invencible funcionando mientras nodos fallan

TÍTULO: "🎯 HITO 2: Te Conviertes en Maestro Anti-Fallo"

EL DESAFÍO: "💣 **REALIDAD DURA:** En producción, servidores fallan TODOS LOS DÍAS Netflix pierde ~100 servidores diarios, Facebook ~1000"




PORQUÉ ES TU SUPERPODER: "🛡️ **MENTALIDAD INVENCIBLE:** Sistemas que se auto-reparan

- DataNode falla → Réplicas automáticas se activan
- NameNode falla → Failover en <60 segundos
- Usuario NUNCA se da cuenta"

CÓMO EVOLUCIONAS: "🧠 **TRANSFORMACIÓN:** De 'espero que no falle' a 'diseño para que falle'"

ARQUITECTURA QUE DOMINAS:






- Replicación inteligente
- Detection automática de fallos
- Recovery sin intervención humana





 **MENSAJE DE ÁNIMO:** "¡IMPRESIONANTE! Ahora dominas tolerancia a fallos como senior engineer. Esta mentalidad te diferencia del 95% de developers. ¡Ya eres inmune al pánico cuando algo falla!  

SLIDE 10: PERFORMANCE CHARACTERISTICS

Imagen de Referencia: Gráficos de performance comparando HDFS vs sistemas tradicionales

TÍTULO: "Performance: Cuándo HDFS Brilla y Cuándo No"



HDFS SWEET SPOT:  **Large Files** (>100MB): Excelente  **Sequential Reads:** Muy rápido
 **Batch Processing:** Perfecto  **Write Once, Read Many:** Ideal  **High Throughput:** Miles de MB/segundo

HDFS LIMITATIONS:  **Small Files** (<1MB): Ineficiente (overhead de metadatos)  **Random Access:** No optimizado para lecturas aleatorias  **Low Latency:** Diseñado para throughput, no latencia 
Many Writers: Un archivo = un writer a la vez

PERFORMANCE NUMBERS:

- **Single Disk:** ~100 MB/s
- **HDFS 10 Nodes:** ~1000 MB/s
- **HDFS 100 Nodes:** ~10,000 MB/s
- **Scale:** Linear hasta miles de nodos

USE CASES:

-  Log processing, ETL, Data warehousing
-  Real-time databases, Gaming, High-frequency trading

SLIDE 11: ARQUITECTURA DE NUESTRO CLUSTER HOY

Imagen de Referencia: Diagrama específico del cluster que construirán

TÍTULO: "Nuestro Cluster HDFS Real de Hoy"

VISUAL ARCHITECTURE:

DOCKER HOST (Tu Laptop)

└─ NameNode Container (Puerto 9870)

└─└─ Metadatos en memoria

└─└─ Web UI para monitoreo

└─└─ Coordina todo el cluster

└─ DataNode1 Container

└─└─ Almacena blocks reales

└─└─ Reporta a NameNode cada 3s

└─└─ Procesa requests de lectura/escritura

└─ DataNode2 Container

└─└─ Réplicas de DataNode1

└─└─ Load balancing automático

└─└─ Fault tolerance

Network: bigdata_network (Docker)

Replication Factor: 2 (perfecto para aprendizaje)

Block Size: 128MB (estándar industria)

REALISTIC SCALE: "2 DataNodes = setup real de startup pequeña"

INTERFACES QUE USAREMOS:

- **http://localhost:9870**: HDFS Web UI
- **Terminal**: Comandos hdfs dfs -ls
- **Jupyter**: Python API para HDFS

SLIDE 12: HITO 3 - CLUSTER HDFS REAL FUNCIONANDO

Imagen de Referencia: Dashboard HDFS mostrando cluster saludable con datos distribuidos

TÍTULO: "🎯 HITO 3: ¡Tu Cluster HDFS Profesional Está VIVO!"

EL DESAFÍO QUE SUPERASTE: "🏆 **LOGRO TÉCNICO:** Cluster HDFS multi-nodo funcionando (Solo 20% de developers han hecho esto)"

PORQUÉ ES ÉPICO: "⚡ **BREAKTHROUGH REAL:** Tienes la misma tecnología que:


- Google usa para indexar internet
- Netflix usa para almacenar petabytes
- Facebook usa para fotos de 3 mil millones de usuarios"

CÓMO EVOLUCIONASTE: "🚀 **TRANSFORMACIÓN COMPLETA:**

- Antes: 'Big Data suena complicado'
- Después: 'Tengo mi propio cluster funcionando''

LO QUE VAS A DOMINAR AHORA:

- Análisis de distribución de datos reales
- Simulación de disaster recovery
- Performance tuning como senior engineer

 **MENSAJE DE ÁNIMO ÉPICO:** "¡FELICITACIONES INGENIERO! 🎉 Acabas de lograr algo que intimida a la mayoría. Tu cluster HDFS está listo para procesar terabytes de datos. ¡Ya eres oficialmente INGENIERO BIG DATA! ¡Vamos a resolver problemas reales de producción! 🌟🚀"

NOTAS PARA EL INSTRUCTOR

Timing Sugerido:

- Slides 1-3: 4 minutos (hook + conceptos básicos)
- Slides 4-6: 4 minutos (arquitectura técnica)
- Slides 7-9: 4 minutos (procesos y fault tolerance)
- Slides 10-12: 3 minutos (performance + transition)

Puntos de Interacción:

- Slide 2: "¿Cuántos usan Google Drive?" (relacionar con experiencia)
- Slide 5: "¿Prefieren más seguridad o más espacio?" (trade-offs)
- Slide 9: "¿Qué harían si su laptop se rompe?" (backup strategies)

Conceptos Críticos:

1. **NameNode vs DataNode** roles claramente diferenciados
2. **Blocks + Replication** = fault tolerance automática
3. **Metadata** es más crítico que los datos mismos
4. **Performance trade-offs** según tipo de workload

Demostración en Vivo:

- Después del slide 12: Abrir <http://localhost:9870>
- Mostrar NameNode UI funcionando
- Ejecutar primer comando (`hdfs dfs -ls /`)
- Transition inmediata a notebooks prácticos

Red Flags a Evitar:

- No profundizar en configuraciones avanzadas
- No explicar internos de algoritmos de consenso
- Enfocarse en USO práctico, no implementación
- Mantener ejemplos relacionados con datos de transporte

Meta: Entender HDFS como herramienta de producción, no como curiosidad académica