Pertemuan 16 — UAS: Tugas Besar

"*Pipeline* data skala besar dengan Apache Spark sampai *dashboard* atau aplikasi", Capaian B

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

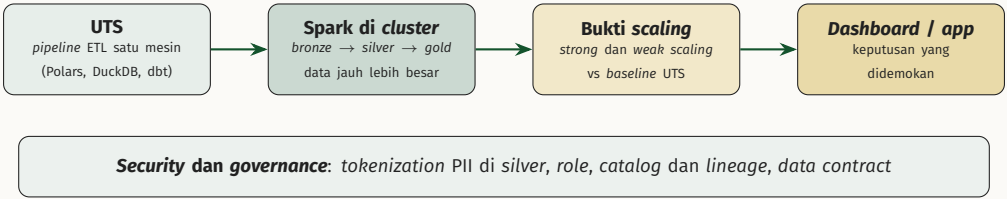
1. Tujuan dan kaitan dengan CPMK
2. Pilihan domain dan data berukuran besar
3. Arsitektur acuan
4. Syarat wajib: Spark, *scaling*, *serving*
5. *Dashboard* atau *data app*
6. *Security* dan *governance* wajib
7. Nilai tambah: *hot path*
8. Skenario demo
9. Repositori, laporan, rubrik, aturan, jadwal
10. Pertanyaan saat demo dan penutup semester

Bobot dan cakupan

UAS bernilai **45%** nilai akhir, mengukur CPMK-4, CPMK-5, dan CPMK-6, dan mencakup materi pertemuan 9–15: Hadoop dan Spark, Spark skala besar, Kafka dan *streaming*, arsitektur *hot* dan *cold*, *dashboard* dan *data app*, *data security*, serta *governance*, GRC, dan *observability*.

Tujuan

Dari satu mesin ke *cluster*, dari tabel ke keputusan



Ringkasan

Sasaran: *bawa pipeline UTS ke skala yang tidak lagi muat dengan nyaman di satu mesin, buktikan dengan angka bahwa cluster memang membantu, lalu sajikan hasilnya sebagai dashboard atau aplikasi yang dipakai untuk mengambil keputusan.*

Kaitan dengan CPMK

CPMK	Kemampuan yang diukur	Bukti di tugas besar
CPMK-4	Spark (RDD, DAG, Spark SQL, *join*, *partitioning*, *tuning*, *scaling*); *streaming* sebagai nilai tambah	*Pipeline* Spark multi-*worker* berjalan, *query plan* dianotasi, kurva *scaling* terukur
CPMK-5	Arsitektur *hot*/*cold*, *serving layer*, *metric*, *dashboard* dan *data app*	Diagram dengan SLO, *serving store* terpilih dengan alasan, *dashboard*/*app* menjawab pertanyaan bisnis
CPMK-6	*Data security*, *governance*/GRC, *observability*	PII ter-*tokenize*, *role* terbukti saat demo, *catalog*, *lineage*, *data contract*

Berbeda dengan UTS (satu mesin), yang dinilai di sini adalah **skala** dan **bukti** bahwa skala itu memang dibutuhkan, sampai ke **penyajian** untuk pengguna.

Pilihan domain dan data

Tiga pilihan domain

Domain	Pertanyaan bisnis di *gold*	*Dashboard* / *app*	PII yang dilindungi
Bank syariah digital (disarankan)	Pembiayaan per akad dan cabang, rasio pembiayaan bermasalah, pola transaksi mencurigakan	*Dashboard* manajemen; *app* pencarian rekening berisiko	NIK, nama, nomor rekening, nomor HP
Transportasi / mobilitas (data terbuka)	Permintaan per zona dan jam, durasi dan tarif rata-rata	Peta panas permintaan; *app* estimasi tarif	ID pengemudi, lokasi presisi
E-commerce	Penjualan per kategori, *cohort* pelanggan, stok	*Dashboard* penjualan; *app* rekomendasi sederhana	Nama, alamat, email

Domain lain boleh diajukan sampai minggu ke-11 bila datanya cukup besar dan memuat data pribadi yang perlu dilindungi.

Data harus cukup besar

```python
# gen/transaksi.py -- generator sintetis yang dapat diskalakan
import numpy as np, pyarrow as pa, pyarrow.parquet as pq
import sys

skala = int(sys.argv[1])          # 1, 2, 4, 8 ...
rng = np.random.default_rng(42)   # seed tetap: demo dapat diulang
baris_per_berkas = 5_000_000
for i in range(10 * skala):       # 10 berkas = 50 juta baris per skala
    n = baris_per_berkas
    t = pa.table({
        "rekening": rng.integers(0, 2_000_000, n),
        "nominal":  rng.lognormal(12, 1.2, n).round(),
        "kanal":    rng.choice(["mobile", "atm", "qris"], n),
        "ts":       rng.integers(1.70e9, 1.73e9, n),
    })
    pq.write_table(t, f"raw/transaksi_{skala}_{i:03d}.parquet")
```

Ukuran minimum (dapat disesuaikan)

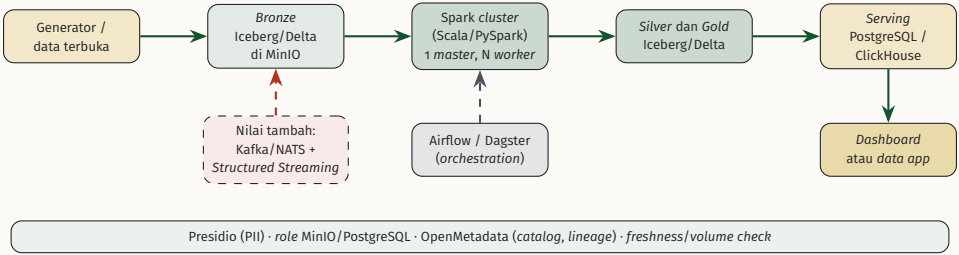
- **Skala 1**: ±50 juta baris transaksi (orde beberapa GB Parquet)
- Eksperimen *weak scaling*: skala 1, 2, 4 bersama 1, 2, 4 *worker*
- Ukuran akhir disepakati di minggu 11 sesuai RAM laptop kelompok

- Alternatif data terbuka: NYC TLC *trip records* (Parquet bulanan), digabung beberapa bulan sampai ukuran setara.
- Data pribadi **selalu sintetis**; tambahkan kolom NIK dan nama palsu (Faker) di tabel dimensi nasabah.

Sumber: Angka di atas adalah titik awal untuk laptop 16 GB dengan *cluster* Docker; tujuan utamanya adalah data yang membuat *baseline* satu mesin mulai kewalahan.

Arsitektur acuan

Arsitektur acuan



Alat boleh diganti asalkan perannya sama dan alasannya ditulis: Delta menggantikan Iceberg, ClickHouse menggantikan PostgreSQL, Streamlit menggantikan Superset, Spark di Kubernetes menggantikan *standalone*. Dasar argumen: tabel *when to use* dari P1 dan P9–P13.

Diagram dan tabel wajib di laporan

Diagram arsitektur

- Semua komponen dan arah aliran data
- Batas tiap *layer*: *bronze*, *silver*, *gold*
- Tempat PII di-*tokenize*
- Jumlah *worker*, *core*, dan memori tiap *executor*

Tabel SLO dan *store*

Produk data	**SLO**	***Store***
Dashboard KPI	$\leq$ 1 jam	PostgreSQL/ClickHouse
Tabel *gold*	harian	Iceberg/Delta
Data mentah	retensi	*bronze*, *cold tier*

Contoh; isi dengan milik Anda dan angka terukur.

Ringkasan

Setiap pilihan dijawab dengan tiga hal: ***latency***, ***cost***, dan **risiko**. “Karena populer” bukan alasan.

Syarat wajib

Syarat wajib 1: *pipeline* Spark

1. *Cluster* Spark dengan **minimal 2 *worker*** (Docker Compose atau Kubernetes lokal).
2. *Job* Spark (Scala atau PySpark) membaca *bronze* dan menulis *silver* serta *gold* ke Iceberg atau Delta di MinIO.
3. *Silver*: *deduplication*, validasi, *conformed type*, *tokenization* PII.
4. *Gold*: minimal satu *fact table* dan dua *dimension*, plus satu tabel agregat untuk *dashboard*.
5. Minimal satu *join* besar dengan strategi yang dipilih secara sadar (*broadcast* atau *sort-merge*) dan satu penanganan *skew*.
6. Ter-*orchestrate* (Airflow atau Dagster) dan idempotent: menjalankan ulang satu tanggal tidak menggandakan data.

Diserahkan sebagai bukti

- Tangkapan *Spark UI*: DAG dan *stage* dengan anotasi di mana *shuffle* terjadi
- *Output* `explain()` *job* utama, dianotasi
- *Configuration executor* dan jumlah *partition* beserta alasannya
- *Data quality test* di *silver* dan *gold* (boleh memakai ulang dari UTS)

Syarat wajib 2: bukti *scaling* (HPC data)

Strong scaling

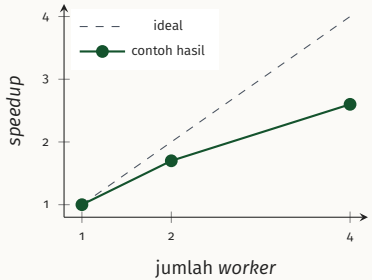
Ukuran data tetap (skala 2 atau 4), jumlah *worker* 1, 2, 4. Laporkan waktu, *speedup* $S = T_1/T_p$, dan efisiensi $E = S/p$ (T_p: waktu dengan p *worker*).

Weak scaling

Data dan *worker* naik bersama ($1\times/1$, $2\times/2$, $4\times/4$). Idealnya waktu tetap; laporkan seberapa jauh dari ideal dan mengapa.

Baseline satu mesin

Query gold yang sama dengan Polars atau DuckDB dari UTS. Di ukuran berapa satu mesin masih menang, dan di ukuran berapa *cluster* mulai unggul?



Kurva di atas hanya **ilustrasi**. Jelaskan penyebab jarak dari ideal: *shuffle*, *skew*, I/O MinIO, *small files*, atau *overhead* JVM. Ulangi tiap pengukuran minimal 3 kali; laporkan median.

Syarat wajib 3: *serving* dan *dashboard* atau *data app*

1. Tabel agregat *gold* dimuat ke *serving store* (PostgreSQL atau ClickHouse) secara terjadwal dan idempotent.
2. Rumuskan **tiga KPI** dengan kartu definisi (rumus, *grain*, filter, pemilik, sumber tabel, frekuensi).
3. Pilih salah satu dan demokan:
   - ***Dashboard*** (Superset atau Metabase): tiga KPI, filter waktu dan cabang/zona, satu *drill-down*.
   - ***Data app*** (Streamlit atau FastAPI + *frontend* sederhana): interaksi yang tidak bisa dilakukan *dashboard* biasa, misalnya pencarian rekening berisiko atau simulasi tarif.
4. Tunjukkan satu keputusan yang dapat diambil dari tampilan itu.

Contoh kartu KPI (bank)

Nama	Rasio pembiayaan bermasalah
Rumus	outstanding kolektibilitas 3–5 / total outstanding
Grain	bulanan, per cabang
Pemilik	divisi risiko
Sumber	`gold.fakta_pembiayaan`

Definisi disederhanakan untuk praktikum; ikuti ketentuan OJK untuk definisi resmi.

Dashboard membaca *serving*, bukan *bronze*

Query dashboard langsung ke tabel mentah ratusan juta baris akan lambat dan membuka akses ke PII.

Security dan *governance* wajib

Kontrol *security* wajib, masing-masing dengan bukti

Kontrol	Minimal	Bukti saat demo
Tokenization PII	NIK, nama, nomor HP di-*tokenize* atau di-*mask* di *silver* (Presidio untuk deteksi)	*Gold* dan *dashboard* hanya menampilkan *token*
Role dan *least privilege*	Minimal dua peran: `engineer` dan `analyst` di MinIO dan *serving store*	`analyst` ditolak membaca *bronze*
Secret	Tidak ada kredensial di Git; `.env` atau Vault	Pencarian riwayat Git bersih
Encryption in transit	TLS pada salah satu jalur (MinIO atau *serving*)	*Configuration* dan koneksi terenkripsi

Ringkasan

Kontrol yang tidak bisa didemokan dianggap tidak ada. Siapkan perintah yang membuktikan setiap baris di atas.

Artefak *governance* wajib

1. ***Catalog* dan *lineage***: tabel *bronze*, *silver*, *gold* terdaftar di OpenMetadata (atau setara) dengan *owner*, deskripsi, *classification*, dan *lineage* sampai *dashboard*.
2. ***Data contract***: minimal satu, antara *gold* dan *dashboard*/*app* (*schema*, semantik, SLO, kebijakan perubahan).
3. **Kebijakan *retention* dan *tiering***: berapa lama tiap *layer* disimpan, di *tier* apa, termasuk *snapshot* Iceberg lama.
4. ***Observability***: *freshness* dan *volume check* pada *gold* dengan *alert*.

Consistency adalah yang dinilai

Artefak yang rapi tetapi tidak sesuai sistem yang berjalan dinilai rendah. Contoh: kebijakan menyebut *bronze* dihapus setelah 90 hari, tetapi tidak ada *lifecycle rule* dan *snapshot* tidak pernah di-*expire*.

Format

Markdown atau YAML di folder `gov/` repositori, sehingga ikut ber-versi bersama kode.

Nilai tambah

Nilai tambah: *hot path* dan lainnya

Hot path (paling dianjurkan)

- *Event* masuk Kafka/Redpanda atau NATS JetStream (generator atau CDC Debezium)
- Spark *Structured Streaming*: *window* berbasis *event time* dengan *watermark*
- Menulis ke *bronze* dan ke *store hot*; tampil di *dashboard*
- *End-to-end latency* P50/P99 terukur

Pilihan lain

- Perbandingan Scala vs PySpark untuk *job* yang sama
- RAPIDS Accelerator (bila ada GPU) dan analisis kapan GPU membantu
- Spark di Kubernetes dengan *dynamic allocation*
- *Risk register* dan matriks UU PDP lengkap
- Uji *right to erasure*: hapus satu nasabah di semua *layer* termasuk *snapshot*
- Estimasi *cost* bulanan di *cloud* (orde besaran)

Skenario demo

Skenario demo wajib (berurutan)

1. Jalankan `docker compose up`; tunjukkan *cluster* Spark dengan semua *worker* sehat.
2. Jalankan DAG: *bronze* → *silver* → *gold* → *serving*; tunjukkan DAG dan *stage* di *Spark UI*.
3. Jalankan ulang satu tanggal; tunjukkan jumlah baris *gold* tidak berubah (idempotent).
4. Tunjukkan grafik *strong* dan *weak scaling* serta perbandingan dengan *baseline* satu mesin, dan jelaskan satu penyebab jarak dari ideal.
5. Buka *dashboard* atau *data app*; jawab satu pertanyaan bisnis secara langsung.
6. Masuk sebagai `analyst`: PII tampil sebagai *token*, akses *bronze* ditolak.
7. Buka *catalog*: telusuri *lineage* satu KPI sampai tabel *bronze*.

Ringkasan

Latih skenario ini sampai bisa berjalan dalam 8 menit. *Job* besar boleh dijalankan sebelum sesi; saat demo cukup jalankan satu *partition* tanggal. Siapkan rekaman layar sebagai cadangan, tetapi demo langsung tetap diutamakan.

Repositori, laporan, dan penilaian

Struktur repositori dan laporan

```
uas-kelompok-XX/
  README.md           # cara jalan
  docker-compose.yml
  .env.example        # tanpa rahasia
  gen/                # generator / unduh data
  spark/              # job Scala/PySpark
  lakehouse/          # DDL tabel
  dags/               # Airflow / Dagster
  serving/            # muat ke PostgreSQL/CH
  dashboard/          # ekspor Superset/Metabase
  app/                # Streamlit/FastAPI
  security/           # role, Presidio
  gov/
    catalog.md  contract.yaml
    retensi.md  checks.yaml
  bench/results/      # CSV pengukuran scaling
  docs/laporan.pdf
  AI-USAGE.md
```

Laporan maksimal 15 halaman:

1. Domain, data, pertanyaan bisnis, dan KPI
2. Arsitektur, SLO, dan alasan pilihan alat
3. *Pipeline* Spark: tiap *layer*, *join*, *skew*, *query plan*
4. Pengukuran *scaling* dan perbandingan dengan *baseline* UTS
5. *Serving* dan *dashboard*/*app*; satu keputusan yang didukung
6. *Security*: kontrol dan buktinya
7. *Governance*: *catalog*, *lineage*, *contract*, *retention*
8. Nilai tambah (bila ada)
9. Keterbatasan dan rencana bila dibawa ke produksi
10. Lampiran: pembagian kerja, catatan penggunaan AI

Presentasi dan demo

Format per kelompok: 20 menit

Presentasi arsitektur dan hasil	6 menit
Demo langsung (skenario wajib)	8 menit
Tanya jawab	6 menit

Pembagian peran saat demo

- Satu orang menjelaskan *pipeline* Spark dan *scaling*
- Satu orang menjelaskan *serving* dan *dashboard*/*app*
- Satu orang menjelaskan *security* dan *governance*
- Semua anggota siap menjawab bagian mana pun

Ringkasan

Nilai individu dapat berbeda dari nilai kelompok bila kontribusi tidak seimbang; riwayat *commit* dan jawaban saat tanya jawab menjadi bahan pertimbangan.

Rubrik penilaian

Komponen	Bobot	Nilai penuh bila
Arsitektur	10%	Diagram lengkap, SLO, alasan *latency–cost*–risiko untuk tiap pilihan
Pipeline Spark	25%	Tiga *layer* benar, multi-*worker*, *join* dan *skew* ditangani, idempotent, *query plan* dianotasi
Bukti *scaling*	15%	*Strong* dan *weak scaling* terukur (median 3 kali), dibandingkan dengan *baseline* UTS, penyebab dijelaskan
Serving dan *dashboard*/*app*	15%	Tiga KPI dengan kartu definisi, tampilan menjawab pertanyaan bisnis, keputusan nyata
Security	10%	Kontrol wajib berjalan dan terbukti saat demo
Governance	10%	Artefak lengkap dan konsisten dengan sistem
Laporan	5%	Struktur lengkap, angka terukur, keterbatasan jujur
Presentasi dan demo	10%	Skenario wajib berjalan; semua anggota menjawab dengan paham
Nilai tambah	*+10%*	*Hot path dengan Structured Streaming, atau pilihan lain di slide nilai tambah*

Total komponen wajib 100%. Nilai akhir UAS dibatasi 100. UAS bernilai 45% dari nilai mata kuliah.

Aturan main dan jadwal

Aturan

- Kelompok 3–4 orang; boleh sama dengan kelompok UTS.
- Data pribadi **selalu sintetis**.
- Alat bantu AI boleh dipakai; tuliskan di `AI-USAGE.md`. Setiap anggota harus mampu menjelaskan kode dan *configuration*-nya.
- Kode atau *configuration* dari sumber lain wajib disebut sumbernya.
- Memalsukan angka pengukuran atau bukti demo: nilai UAS nol.
- Bila bersinggungan dengan tugas besar mata kuliah lain, masing-masing dinilai terpisah sesuai rubriknya.
- Keterlambatan dan tenggat laporan mengikuti ketentuan yang diumumkan di kelas.

Jadwal (pertemuan Senin)

P9 (30 Nov)	Tugas diumumkan; kelompok dan domain
P10 (7 Des)	*Cluster* Spark dan data skala 1 jalan
P11 (14 Des)	*Silver* dan *gold* pertama
P12 (21 Des)	**Rancangan arsitektur dinilai**
P13 (28 Des)	*Serving* dan *dashboard*/*app*
P14 (4 Jan)	Kontrol *security*; pengukuran *scaling*
P15 (11 Jan)	*Governance*; **presentasi progres**
P16 (18 Jan)	**Presentasi akhir dan demo**

Tanggal mengikuti kalender mingguan; sesuaikan bila ada libur.

Checklist pengumpulan

- ☐ Tautan repositori dapat diakses dosen
- ☐ `README.md`: cara menjalankan dari nol
- ☐ `docker compose up` berjalan bersih
- ☐ Generator dengan *seed* atau skrip unduh data
- ☐ *Job* Spark dan DAG
- ☐ CSV pengukuran *scaling* dan *baseline*
- ☐ Kartu definisi tiga KPI
- ☐ Ekspor *dashboard* atau kode *app*
- ☐ Bukti kontrol *security*
- ☐ Artefak di `gov/`
- ☐ Laporan PDF $\leq$ 15 halaman, `AI-USAGE.md`
- ☐ *Slide* presentasi

Pertanyaan yang akan muncul saat demo

1. Di mana *shuffle* terjadi pada *job* utama Anda, dan bagaimana Anda menguranginya?
2. Mengapa *join* itu memakai *broadcast* (atau tidak)? Apa yang terjadi bila tabel dimensinya membesar sepuluh kali?
3. *Speedup* Anda di 4 *worker* jauh dari 4. Apa penyebab utamanya, dan apa buktinya?
4. Di ukuran data berapa Polars atau DuckDB di satu mesin masih lebih cepat? Kapan Anda akan menyarankan tidak memakai Spark?
5. Menjalankan ulang *job* kemarin: mengapa hasilnya tidak dobel?
6. Seorang nasabah meminta datanya dihapus. Di mana saja datanya berada, termasuk *snapshot* Iceberg dan *serving store*?
7. Mengapa *dashboard* (atau *app*) itu, dan keputusan apa yang dibantunya?
8. Tunjukkan satu *configuration* atau baris kode yang Anda tulis sendiri dan jelaskan.

Menutup semester

Paruh pertama: membangun *pipeline* ETL yang **benar, idempotent, dan murah** di satu mesin, dari *data modeling*, *storage*, *lakehouse bronze–silver–gold*, Polars dan DuckDB, sampai *orchestration*.

Paruh kedua: membawanya ke **skala besar** dengan Spark, menambah *streaming*, menyajikannya sebagai *dashboard* dan aplikasi, lalu menjaganya agar **aman, patuh, dan terpantau**.

Platform data yang baik membuat keputusan lebih cepat dan lebih benar, dengan data yang dijaga sebagai amanah. Jumlah alat yang dipakai tidak menjadi ukurannya.

Bekal ke depan

Bawa tiga pertanyaan ini ke setiap rancangan: **seberapa besar dan segar** datanya, **berapa *cost*-nya**, dan **apa risikonya** bila salah atau bocor. Lalu **ukur**, jangan menebak.

Selamat mengerjakan

Terima kasih untuk satu semester ini